DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

Approved for public release; distribution is unlimited

Design of a GPS Aided Guidance, Navigation, and Control System for Trajectory Control of an Air Vehicle

by

Eric N. Hallberg
Lieutenant, United States Navy
B.S. University of Pennsylvania, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

March , 1994

Department of Aeronautics and Astronautics

REPORT DOCUMENTATION PAGE

a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(if applicable)* AA	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943
a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(if applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNI ACCESSION

. TITLE *(Include Security Classification)*

Design of a GPS Aided Guidance, Navigation, and Control System for Trajectory Control of an Air Vehicle

. PERSONAL AUTHOR(S)

LT. Eric N. Hallberg

a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM 01/92 TO 03/94	14. DATE OF REPORT *(Year, Month, Day)* March 1994	15. PAGE COUNT 119

. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the fficial policy or position of the Department of Defense or the United States Government.

COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Trajectory Control, GNC, Linear Quadratic Regulator, LQR, Kalman Filt Nonlinear Simulation, $\mathcal{D}$-Implementation, Bluebird

. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The advent of GPS has afforded the aerospace controls engineer a powerful, new means of controlling air vehic his work explores a new method of designing and implementing controllers and guidance systems for autonomous con air vehicles utilizing a GPS integrated guidance, navigation and control system. This is a subject of considera terest when realizing controllers to track reference trajectories given in an inertial reference frame. The desi plementation, and dynamic simulation of a precise tracking trajectory controller for an Unmanned Air Vehicle (U presented. This design provides a natural conversion of commands and other measured outputs (such as GPS sign om an inertial reference frame to a body-fixed reference frame. This achieves automatic recruiting of the actua hile preserving the properties of the original design (linearization principle).

. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
a. NAME OF RESPONSIBLE INDIVIDUAL Isaac I. Kaminer	22b. TELEPHONE *(Include Area Code)* (408) 656-2972	22c. OFFICE SYMBOL AA/KA

FORM 1473, 84 MAR — 83 APR edition may be used until exhausted
All other editions are obsolete

ABSTRACT

The advent of GPS has afforded the aerospace controls engineer a powerful, new means of controlling air vehicles. This work explores a new method of designing and implementing controllers and guidance systems for autonomous control of air vehicles utilizing a GPS integrated guidance, navigation and control system. This is a subject of considerable interest when realizing controllers to track reference trajectories given in an inertial reference frame. The design, implementation, and dynamic simulation of a precise tracking trajectory controller for an Unmanned Air Vehicle (UAV) is presented. This design provides a natural conversion of commands and other measured outputs (such as GPS signals) from an inertial reference frame to a body-fixed reference frame. This achieves automatic recruiting of the actuators while preserving the properties of the original design (linearization principle).

H16125
c.1

TABLE OF CONTENTS

I. INTRODUCTION 1
II. DEVELOPMENT OF THE DYNAMIC MODEL 4
  A. REFERENCE FRAMES 4
    1. Local Tangent Plane Reference Frame 4
    2. Body-Fixed Reference Frame 5
    3. Flight Path or Wind Reference Frame 6
  B. COORDINATE TRANSFORMATIONS 7
  C. NOTATION 9
  D. RIGID BODY EQUATIONS OF MOTION 10
    1. Linear Motion 10
    2. Angular Rotation 11
    3. External Forces and Moments 12
    4. The State Space Representation 14
    5. Trim and Linearization 19
III. THE LINEAR QUADRATIC REGULATOR DESIGN 23
  A. LQR OVERVIEW 23
  B. DESIGN REQUIREMENTS 27
  C. THE SYNTHESIS MODEL 28
  D. THE DESIGN PROCESS 32
  E. LQR CONTROLLER PERFORMANCE 34
IV. CONTROLLER IMPLEMENTATION ON THE NONLINEAR PLANT 41
  A. $\mathcal{D}$-IMPLEMENTATION 41
  B. $\mathcal{D}$-IMPLEMENTATION OF THE CONTROLLER IN SIMULINK 47

MONTEREY CA 93943-5101

C. GENERATION OF THE TRAJECTORY COMMANDS 48
D. STATE FEEDBACK TO OUTPUT FEEDBACK 54
1. Sensor Modeling 54
2. Kalman Filtering 58
E. INTEGRATION OF THE FULL NONLINEAR SIMULATION . . 60
V. APPLICATION TO THE CONTROL OF BLUEBIRD 63
A. $\mathcal{D}$-IMPLEMENTED CONTROLLER PERFORMANCE CHARACTERISTICS 63
B. AN AIRPORT DEPARTURE AND ARRIVAL FLIGHT SIMULATION 72
VI. CONCLUSIONS AND RECOMMENDATIONS 78
A. CONCLUSIONS 78
B. RECOMMENDATIONS 79
APPENDIX A: MATLAB FILES 81
APPENDIX B: SIMULINK FILES 98
APPENDIX C: $\mathcal{D}$-IMPLEMENTATION PROOFS REFERENCED . . . 100
REFERENCES 106
INITIAL DISTRIBUTION LIST 107

LIST OF TABLES

3.1 EIGENVALUES OF BLUEBIRD 29
3.2 EIGENVALUES OF BLUEBIRD WITH YAW DAMPER 30
3.3 TRANSMISSION ZEROS OF SYNTHESIS MODEL 34
3.4 EIGENVALUES OF THE FEEDBACK SYSTEM 35
3.5 COMMAND AND CONTROL BANDWIDTHS 38
4.1 ACCELEROMETER CHARACTERISTICS 55
4.2 ANGULAR RATE SENSOR CHARACTERISTICS 56
4.3 INCLINOMETER AND MAGNETOMETER CHARACTERISTICS 56

LIST OF FIGURES

2.1 Local Tangent Plane Coordinate System 5

2.2 Body-Fixed Coordinate System 6

2.3 Wind or Flight Path Reference Frame 7

2.4 SIMULINK Nonlinear Sixteen State Dynamic Model of an Air Vehicle 20

2.5 SIMULINK Nonlinear Nine State Dynamic Model of an Air Vehicle . 21

3.1 Standard LQR feedback configuration. 24

3.2 Feedback Configuration for Root Locus Analysis. 27

3.3 Synthesis and Analysis Model 31

3.4 Control-Loop Bandwidth: Elevator Channel 35

3.5 Control-Loop Bandwidth: Throttle Channel 36

3.6 Control-Loop Bandwidth: Aileron Channel 36

3.7 Command-Loop Bandwidth: X Position Channel 37

3.8 Command-Loop Bandwidth: Y Position Channel 37

3.9 Command-Loop Bandwidth: Z Position Channel 38

3.10 Bank Angle and Heading Response to Ramp in Y Command 39

3.11 Pitch Angle and Altitude Response to Ramp in Z Command 39

3.12 Trajectory Error Due to a Constant Wind Disturbance 40

4.1 Block diagram of the nonlinear controller $\mathcal{K}\ (\Lambda)$ 46

4.2 $\mathcal{D}$-Implementation of controller on Bluebird 49

4.3 Commanded Trajectory Logic Block 50

4.4 Example of Commanded Trajectory Revision 52

4.5 Generation of Trajectory Commands 53

4.6 Angular Rate Sensor 55

4.7 Sensor Models in SIMULINK 57
4.8 Frequency Response of Position Filter 59
4.9 Frequency Response of Euler Angle Filter 60
4.10 SIMULINK Diagram of the Full Nonlinear Simulation 62
5.1 Test Trajectory #1 65
5.2 Trajectory #1: Position Error and Euler Angles 66
5.3 Trajectory #1: Velocity Data 67
5.4 Trajectory #1: Control Activity 68
5.5 Test Trajectory #2 69
5.6 Trajectory #2: Position Error and Euler Angles 70
5.7 Trajectory #2: Velocity Data 71
5.8 Trajectory #2: Control Activity 72
5.9 Trajectory #2: Position Error for Varying Turn Rates 73
5.10 Trajectory #2: Position Error for Varying Wind Velocities 74
5.11 Departure and Arrival at an Airfield 75
5.12 Airfield Scenario: Position Error and Euler Angles 76
5.13 Airfield Scenario: Control Activity 77

ACKNOWLEDGMENT

There are a few special people I would like mention without whom this project would not have been completed. It was a true pleasure to have studied aeronautics under Professor Isaac Kaminer and Professor Rick Howard. Their advice and professional counsel were invaluable. I am especially grateful to Professor Isaac Kaminer whose vision and expertise are directly responsible for the creation of the superb avionics lab where this work was accomplished. Finally, a special thanks to five individuals who sacrificed the most. To Nicole, Katie, Eric, and Patty, you were the best while understanding the least. And to Patricia, thank you for your exceptional patience; without you I would not have been able to complete this project.

I. INTRODUCTION

The advent of GPS has afforded the aerospace controls engineer a powerful new means of controlling air vehicles. Current guidance schemes rely in some part on ground based radars, navigational aides, beacons, localizer beams, etc. Guidance and control of the air vehicle have been developed seperately and then combined in a somewhat adhoc process. Precise tracking of inertial fixed trajectories using an onboard GPS integrated GNC suite affords a quantum leap in autonomous flight capabilities. The greatest impact of the proposed technology is expected in the area of trajectory tracking control for autonomous unmanned vehicles, and automatic approach and landing of manned vehicles dicussed next.

Control of the commercial air traffic throughout the country continues to become more and more demanding as an increased number of vehicles vie for limited access to the major commercial aviation hubs. Sophisticated and expensive ground based radar control facilities employ a large number of personnel to individually instruct the pilots of these aircraft on the trajectories they are to fly. Precise control of the aircraft trajectory such as required by an approach into an airfield requires constant attention from a ground based air traffic controller. Furthermore, ATC ability to effectively control the aircraft is influenced by ground based radar coverage and navaid equipment available and is often negatively influenced by atmospheric conditions. Airfields with limited resources are often unable to take-off and land aircraft requiring instrument departures and arrivals.

Flight patterns around major aviation hubs are, in general, inertial based trajectories. That is, an aircraft is required to track a certain path over the ground while adhering to a certian altitude schedule, irrespective of air mass disturbances. In some cases, such as on final approach to land or when the aerodrome is situated among significant terrain or cultural development, precise adherence to the desired trajectory is crucial for flight safety. In any case, commanding an inertial trajectory directly utilizing a GPS integrated GNC system could prove to be more cost effective and accurate than current methods. Furthermore, such a guidance scheme could open up many more airports to significant commercial air traffic without requiring the capital investment and maintenance of ground based radar facilities.

Unmanned air vehicles can be a cost effective means of power projection. Additionally, in some cases human physiological limits may prove to be the limiting factor in the performance of an air vehicle. The precision delivery of munitions becomes of paramount importance as weapons and weapon delivery platforms continue to increase in cost, thus limiting their numbers. All of these concerns can be addressed by autonomous air vehicles utilizing a GPS aided guidance, navigation and control suite.

All of these applications have a common thread running through them. While the particulars of the vehicles may vary considerably, the intent is to acheive autonomous control of their trajectory. As a proof of concept, this work presents a new design process for the synthesis of a guidance, navigation, and control system for a UAV named Bluebird. The function of the this GNC system is to track inertial trajectories. Bluebird is a UAV operated at the Unmanned Air Vehicle Lab at the Naval Postgraduate School. It has a 12.5 foot wingspan and a 20 pound payload capability, and is currently being equipped with a full avionics suite, including IMU, GPS, and air data sensors.

The design process began with the development of a nonlinear dynamic model of Bluebird implemented in SIMULINK. A typical cruise flight condition was chosen as the point for linearization. After linearization of the nonlinear model, the work centered on the design of a linear controller. LQR (Linear Quadratic Regulator) synthesis approach was used since it provides an intuitive means of synthesizing a multivariable controller within the framework of real world design constraints. Following the design of the LQR controller, the challenge of implementing the linear controller on the nonlinear plant was addressed. A novel method of converting commands and outputs from inertial to body reference cooridinates was used [Ref. 10]. This method achieves automatic recruiting of the actuators, while preserving a certain linearization property. Next, the accuracy of the nonlinear simulation was enhanced with the addition of high fidelity models of sensors used onboard Bluebird. Additionally, Kalman filters were designed in order to provide optimal state estimates. Finally, the performance of the controller was evaluated in simulations with the full nonlinear model.

II. DEVELOPMENT OF THE DYNAMIC MODEL

The development of an integrated guidance, navigation, and control system required a high fidelity nonlinear model of the aircraft dynamics. The discussion begins with an explanation of nomenclature, abbreviations, and a definition of frames of reference.

A. REFERENCE FRAMES

Three different reference frames are used in this report. They are:

- Local Tangent Plane or Inertial Reference Frame
- Body-Fixed Reference Frame
- Wind or Flight Path Reference Frame

1. Local Tangent Plane Reference Frame

The position of the air vehicle must be maintained with respect to the local tangent plane coordinate system. This coordinate system is formed by extending a ray from the center of the earth to its surface. A plane is attached tangent to the point of intersection of the ray with the Earth's surface. While it is somewhat arbitrary, for our purposes here it will be convenient to define the positive x direction as pointing east, the positive y direction as pointing north, and the positive z direction as pointing up. This is depicted in Figure 2.1.

For the purposes of this development, the rotation of the earth and its associated Coriolis' forces can be ignored and the local tangent plane reference frame

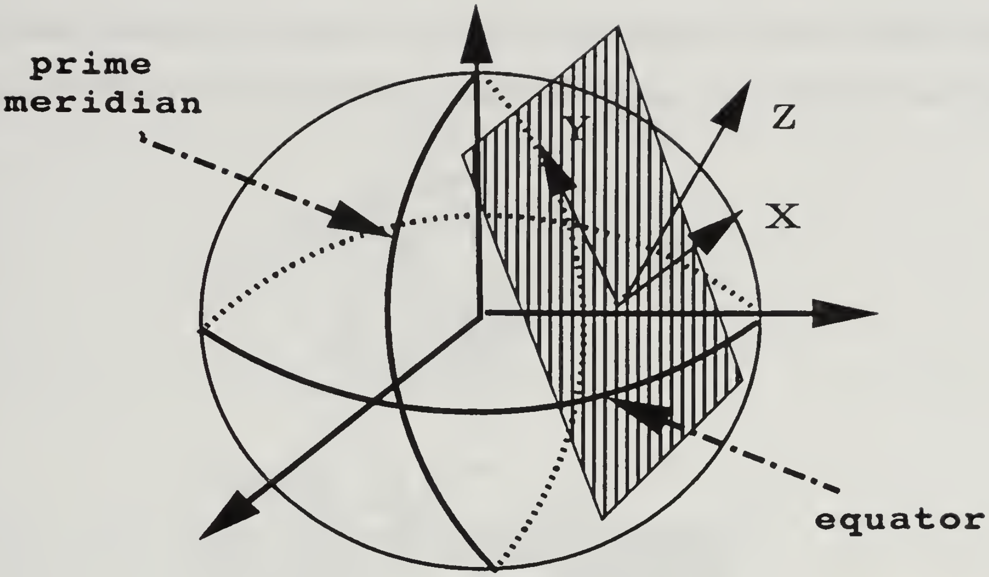
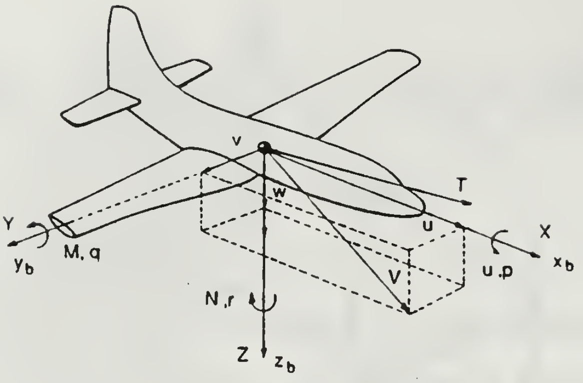


Figure 2.1: Local Tangent Plane Coordinate System

can be considered to be an inertial reference frame. In this work, $\{I\}$ is used to represent the inertial reference frame.

2. Body-Fixed Reference Frame

The body-fixed reference frame is a right hand orthogonal system with the origin at the center of gravity of the air vehicle. The positive x direction points towards the nose. The positive y direction points out the right wing and the positive z direction points towards the bottom of the air vehicle. The velocity of the air vehicle with respect to the inertial reference frame, resolved along the x, y, and z axis of the body-fixed reference frame, are termed *u, v,* and *w*, respectively. The angular rate of rotation of the air vehicle with respect to the inertial reference frame, resolved in the body-fixed reference frame, are called *p, q,* and *r*, respectively. Positive values for the forces, moments, angular rates, and linear velocities in the body-fixed reference frame are shown in Figure 2.2. The abrieviation, $\{B\}$, is used to represent the body-fixed reference frame.



	Roll Axis x_b	Pitch Axis y_b	Yaw Axis z_b
Angular rates	p	q	r
Velocity components	u	v	w
Aerodynamic force components	X	Y	Z
Aerodynamic moment components	L	M	N
Moment of inertia about each axis	I_x	I_y	I_z
Products of inertia	I_{yz}	I_{xz}	I_{xy}

Figure 2.2: Body-Fixed Coordinate System [Ref. 11]

3. Flight Path or Wind Reference Frame

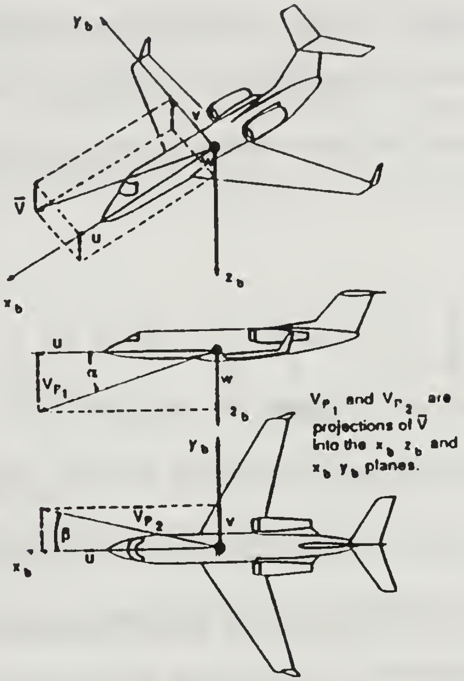
The wind reference frame is also a right hand orthogonal system with its origin at the center of gravity, c.g., of the air vehicle. The x axis is aligned with the velocity vector of the air vehicle. The orientation of the wind reference frame with respect to the body-fixed reference frame is defined in terms of the angles α and β. The equations for α and β are given below.

$$\alpha = tan^{-1}(w/u) \tag{2.1}$$

and

$$\beta = sin^{-1}(v/V) \tag{2.2}$$

where the vectors u, v, w, and V are velocity components of the air vehicle defined in Figure 2.3. The abrieviation, $\{W\}$, is used to represent the wind reference frame.



V_{P_1} and V_{P_2} are projections of $\bar{V}$ into the x_b z_b and x_b y_b planes.

Figure 2.3: Wind or Flight Path Reference Frame [Ref. 11]

B. COORDINATE TRANSFORMATIONS

In order to use these three coordinate systems, one must be able to transform between them freely. The Euler angles, Φ, Θ, and Ψ, termed roll, pitch, and yaw, are defined in order to express the orientation of the body-fixed reference frame with respect to the inertial reference frame. For the purposes of this development, a 3-2-1 Euler angle transformation will suffice as its singularity occurs at Θ equal to 90 degrees. The 3-2-1 transformation is given without explanation but a good development of Euler angle transformations in general can be found in [Ref. 6]. The nature of the angular rotation is more apparent when the transformation is expressed as the

$$\frac{\partial C}{\partial \Delta} = \begin{bmatrix} C_{L_{\delta_e}} & C_{L_{\delta_r}} & C_{L_{\delta_a}} \\ C_{Y_{\delta_e}} & C_{Y_{\delta_r}} & C_{Y_{\delta_a}} \\ C_{D_{\delta_e}} & C_{D_{\delta_r}} & C_{D_{\delta_a}} \\ C_{l_{\delta_e}} & C_{l_{\delta_r}} & C_{l_{\delta_a}} \\ C_{m_{\delta_e}} & C_{m_{\delta_r}} & C_{m_{\delta_a}} \\ C_{n_{\delta_e}} & C_{n_{\delta_r}} & C_{n_{\delta_a}} \end{bmatrix},$$

where x' is the vector composed of u/U, α, β, p, q, and r. Now the aerodynamic forces and moments can be expressed as follows:

$$\begin{bmatrix} {}^{B}F_{AERO} \\ {}^{B}N_{AERO} \end{bmatrix} = \bar{q}\,\bar{S} \begin{bmatrix} {}^{B}_{W}R & 0 \\ 0 & {}^{B}_{W}R \end{bmatrix} \{C_{F_0} + \frac{\partial C}{\partial x'} M' x + \frac{\partial C}{\partial \dot{x}'} \dot{M}' \dot{x} + \frac{\partial C}{\partial \Delta} \Delta\}, \qquad (2.22)$$

where M', $\bar{q}$, and $\bar{S}$ are matrices used to dimensionalize the stability and control derivatives and convert the state vector, x, to x':

$$x = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T,$$

$$\bar{q} = dynamic\ \ pressure,$$

$$\bar{S} = \mathrm{diag}\{-S, S, -S, Sb, Sc, Sb\},$$

$$x' = M'x,$$

$$M' = \mathrm{diag}\{1/V_T, 1/V_T, 1/V_T, b/2V_T, c/2V_T, b/2V_T\},$$

and

$$\dot{x}' = \dot{M}'\dot{x},$$

$$\dot{M}' = \mathrm{diag}\{0, c/(2V_T), b/(2V_T), 0, 0, 0\}.$$

Equation 2.16 can now be further expanded using expressions of the forces and moments derived above.

$$\frac{d}{dt}\begin{bmatrix} {}^{B}v_{cg} \\ {}^{B}\omega_{B} \end{bmatrix} = \begin{bmatrix} -{}^{B}\omega_{B}\times & 0 \\ 0 & -{}^{B}J_{B}^{-1}({}^{B}\omega_{B}\times {}^{B}J_{B}{}^{B}\omega_{B}) \end{bmatrix}\begin{bmatrix} {}^{B}v_{cg} \\ {}^{B}\omega_{B} \end{bmatrix} + \begin{bmatrix} \frac{I}{m} & 0 \\ 0 & {}^{B}J_{B}^{-1} \end{bmatrix}\begin{bmatrix} {}^{B}F \\ {}^{B}N \end{bmatrix}, \tag{2.23}$$

where

$$\begin{bmatrix} {}^{B}F \\ {}^{B}N \end{bmatrix} = \left\{ \begin{bmatrix} {}^{B}F_{GRAV} \\ 0 \end{bmatrix} + \begin{bmatrix} {}^{B}F_{PROP} \\ 0 \end{bmatrix}\delta_{trt} + \left\{ \begin{bmatrix} {}^{B}_{W}R & 0 \\ 0 & {}^{B}_{W}R \end{bmatrix} \cdot \bar{q}\bar{S}\left\{ C_{FO} + \tfrac{\partial C}{\partial x'}M'x + \tfrac{\partial C}{\partial \dot{x}'}\dot{M}'\dot{x} + \tfrac{\partial C}{\partial \Delta}\Delta \right\} \right\} \right\}. \tag{2.24}$$

Notice that there is a state derivative term on the right hand side of Equation 2.23 due to the second order terms in the Taylor series expansion of the aerodynamic forces and moments in Equation 2.24. By bringing it to the left hand side of Equation 2.24 and combining terms, we obtain:

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix} {}^{B}v_{cg} \\ {}^{B}\omega_{B} \end{bmatrix} = \chi^{-1}\Bigg\{ \Bigg[\begin{bmatrix} -{}^{B}\omega_{B}\times & 0 \\ 0 & -{}^{B}J_{B}^{-1}({}^{B}\omega_{B}\times {}^{B}J_{B}{}^{B}\omega_{B}) \end{bmatrix} &+ \\ M_{I}^{-1}{}^{B}_{W}T\bar{q}\bar{S}\tfrac{\partial C_{F}}{\partial x'}M' \Bigg] \begin{bmatrix} {}^{B}v_{cg} \\ {}^{B}\omega_{B} \end{bmatrix} + M_{I}^{-1}\Bigg\{ \begin{bmatrix} {}^{B}F_{GRAV} \\ 0 \end{bmatrix} &+ \\ \begin{bmatrix} {}^{B}F_{PROP} \\ 0 \end{bmatrix}\delta_{trt} + {}^{B}_{W}T\bar{q}\bar{S}(C_{F_0} + \tfrac{\partial C_{F}}{\partial \Delta}\Delta) \Bigg\} \Bigg\}&, \end{aligned} \tag{2.25}$$

where:

$${}^{B}_{W}T = \begin{bmatrix} {}^{B}_{W}R & 0 \\ 0 & {}^{B}_{W}R \end{bmatrix} \text{ and } M_{I} = \begin{bmatrix} m & 0 \\ 0 & {}^{B}J_{B} \end{bmatrix}$$

and

$$\chi = I_{6} - M_{I}^{-1}{}^{B}_{W}T\bar{q}\bar{S}\frac{\partial C_{F}}{\partial \dot{x}'}\dot{M}'. \tag{2.26}$$

Equation 2.25 expresses the derivative of the first six states of the nonlinear model in matrix form. It is solved in the user defined *MATLAB Fcn* block, *state-deriv.m*, which is available for inspection in Appendix A. The physical parameters specific to Bluebird are stored in a *MATLAB .m* file, *Bluebird-data.m*, and are called from within the function, *state-deriv.m* . In order to change the vehicle being modeled, one simply changes the constants in *Bluebird-data.m* .

Next, the Euler angles were added as the additional three states of the model. The time history of the Euler angles is defined in Equation 2.4 and written in compact form as:

$$\dot{\Lambda} = S(\Lambda)^{B}\omega_B, \tag{2.27}$$

where

$$\Lambda = \begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix},$$

and

$$S(\Lambda) = \begin{bmatrix} 1 & \sin\Phi \tan\Theta & \cos\Phi \tan\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi \sec\Theta & \cos\Phi \sec\Theta \end{bmatrix}.$$

The user defined *MATLAB Fcn* block, *eul.m*, solves Equation 2.27, the details of which can be found in Appendix A.

Finally, the inertial position of Bluebird can be computed as follows.

$$^{I}\dot{P}_{cg} = ^{I} v_{cg} = {}^{I}_{B}R^{B}v_{cg}. \tag{2.28}$$

The rotation matrix, ${}^{I}_{B}R$, is implemented in a *MATLAB Fcn* block, *posb2i.m,* in order to convert the vector $^{B}v_{cg}$ to the vector $^{I}v_{cg}$. The vector $^{I}v_{cg}$ is then integrated to obtain a time history of the position of the air vehicle in the local tangent plane. To increase fidelity of the simulation, a first order model of the four actuators, $\delta_{elevator}$, δ_{rudder}, $\delta_{aileron}$, $\delta_{throttle}$, is included with a time constant of 1/12. The resulting nonlinear dynamic model now has sixteen states which are computed using Equations 2.25, 2.27,and 2.28; summarized here for clarity:

$$
\begin{aligned}
states &= \begin{bmatrix} u & v & w & p & q & r & \Phi & \Theta & \Psi & Xpos & Ypos & Zpos & \delta_{e_{pos}} & \delta_{r_{pos}} & \delta_{a_{pos}} & \delta_{t_{pos}} \end{bmatrix}^T \\
&= \begin{bmatrix} {}^B v_{cg}^T & {}^B\omega_B^T & \Lambda^T & P^T & Act_{pos}^T \end{bmatrix}^T,
\end{aligned}
$$

$$
\begin{aligned}
\frac{d}{dt}\begin{bmatrix} {}^B v_{cg} \\ {}^B\omega_B \end{bmatrix} = \chi^{-1}\Bigg\{ \Bigg[\begin{bmatrix} -{}^B\omega_B \times & 0 \\ 0 & -{}^B J_B^{-1}({}^B\omega_B \times {}^B J_B {}^B\omega_B) \end{bmatrix} & + \\
M_I^{-1} {}^B_W T \bar{q} \bar{S} \tfrac{\partial C_F}{\partial x'} M' \Bigg] \begin{bmatrix} {}^B v_{cg} \\ {}^B\omega_B \end{bmatrix} + M_I^{-1} \Bigg\{ \begin{bmatrix} {}^B F_{GRAV} \\ 0 \end{bmatrix} & + \\
\begin{bmatrix} {}^B F_{PROP} \\ 0 \end{bmatrix} \delta_{trt} + {}^B_W T \bar{q} \bar{S} (C_{F_0} + \tfrac{\partial C_F}{\partial \Delta} \Delta) \Bigg\} \Bigg\}, & \qquad (2.29)
\end{aligned}
$$

$$\dot{\Lambda} = S(\Lambda) {}^B\omega_B, \qquad (2.30)$$

$${}^I\dot{P}_{cg} = {}^I_B R {}^B v_{cg}. \qquad (2.31)$$

The SIMULINK diagram of the nonlinear model is shown in Figure 2.4.

5. Trim and Linearization

For linear controller design, the nonlinear equations above must be trimmed and linearized for a typical cruise flight condition for Bluebird. A SIMULINK tool is available which can be used to find equilibrium points of nonlinear dynamic models. The user specifies which states and control inputs are to remain fixed along with their stationary values and the trim routine searches for values of the state and input vector for which the derivative of the state vector equals zero. With the trim condition known, another SIMULINK tool perturbs the states around the specified trim point in order to find the rate of change of the states and control inputs (Jacobian). The resulting linear model is returned in state space format. Since Equation 2.28 can only

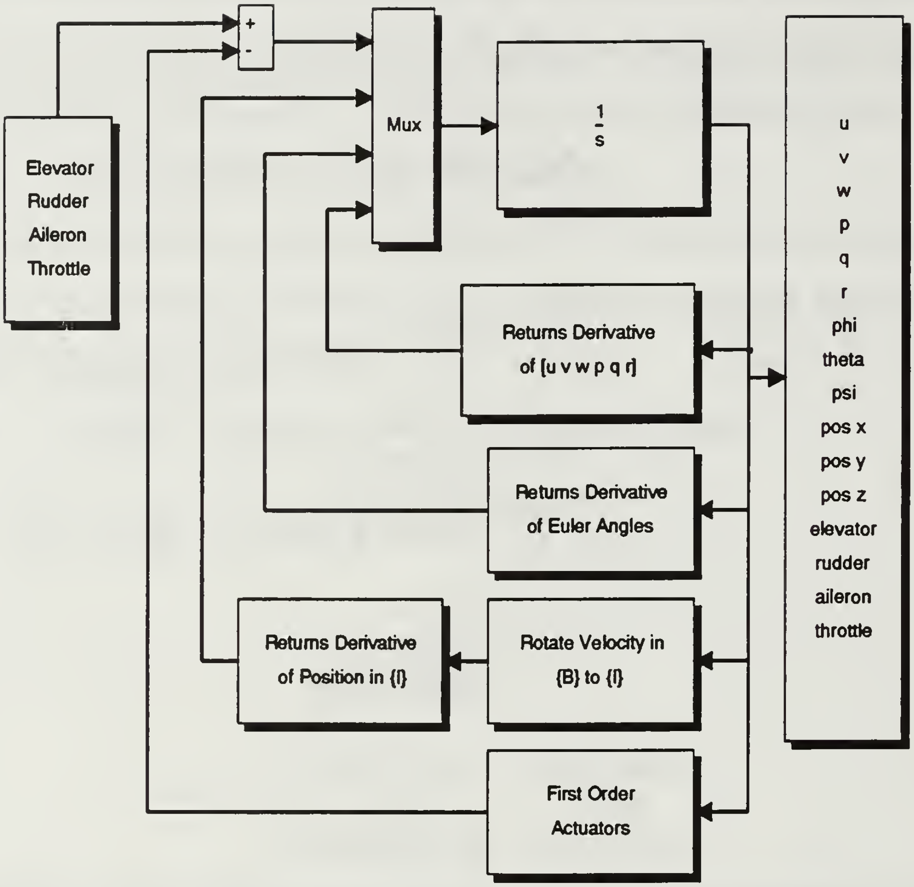

+
-
Elevator
Rudder
Aileron
Throttle
Mux
1/s
u
v
w
p
q
r
phi
theta
psi
pos x
pos y
pos z
elevator
rudder
aileron
throttle
Returns Derivative of [u v w p q r]
Returns Derivative of Euler Angles
Returns Derivative of Position in {I}
Rotate Velocity in {B} to {I}
First Order Actuators

Figure 2.4: SIMULINK Nonlinear Sixteen State Dynamic Model of an Air Vehicle

be trimmed for $v_{cg} = 0$, not a typical flight condition, we will trim Equations 2.25 and 2.27, and then include Equation 2.28 for linearization.

We are interested in trimming the model in velocity. Hopefully, the derivative of the position states will never equal zero in flight. Also, in trim, the control inputs are equal to the actuator positions. Therefore, actuator states are also removed from the nonlinear model. The nine state nonlinear model of Bluebird used for trim is shown in Figure 2.5.

A typical cruise flight condition for Bluebird is given by:

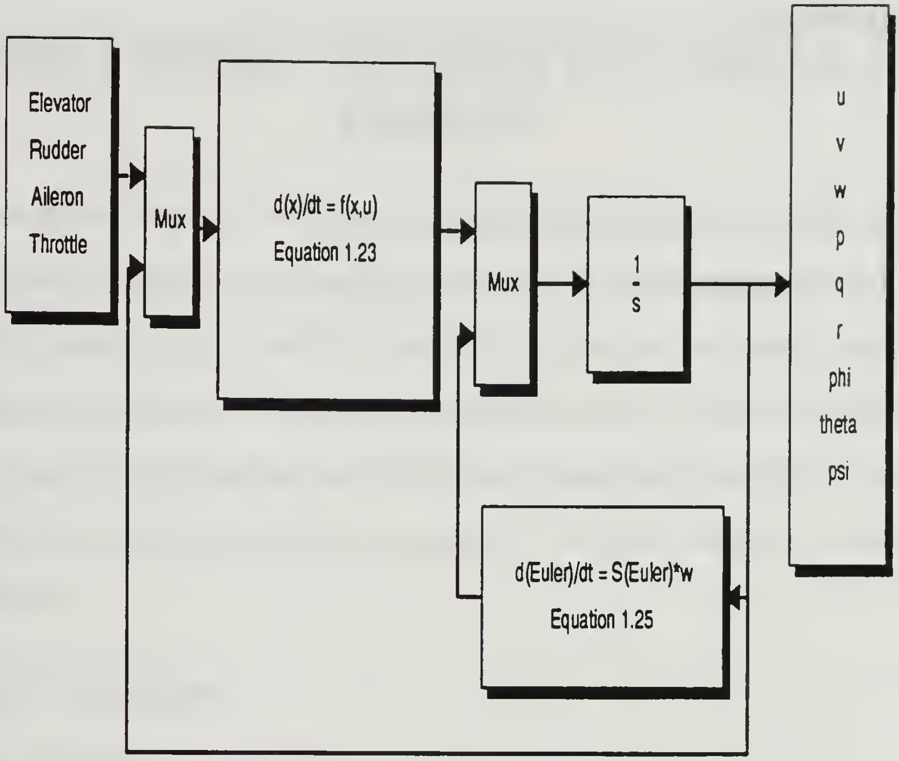

Elevator
Rudder
Aileron
Throttle
Mux
d(x)/dt = f(x,u)
Equation 1.23
Mux
1
-
s
u
v
w
p
q
r
phi
theta
psi
d(Euler)/dt = S(Euler)*w
Equation 1.25

Figure 2.5: SIMULINK Nonlinear Nine State Dynamic Model of an Air Vehicle

- flight speed equal to 73 feet per second
- flight path angle equal to zero
- wings level attitude

The nonlinear model depicted in Figure 2.4 was trimmed at this condition. The trim values of the nine states, u, v, w, p, q, r, Φ, Θ, and Ψ, and the four control inputs, δ_e, δ_r, δ_a, and δ_t were returned. While the sixteen state nonlinear model cannot be trimmed in position, it can be linearized at an arbitrary position. The origin of $\{I\}$ is conveniently chosen. These values were then used to linearize the complete nonlinear model, including position and actuator states. The resulting linear model of Bluebird and numerical values for the trim condition are included in the Appendix B.

III. THE LINEAR QUADRATIC REGULATOR DESIGN

The previous chapter developed a nonlinear model of Bluebird, which was used to derive a linear model for a cruise flight condition. In this chapter, a linear dynamic controller is developed to provide trajectory tracking for the linear model. LQR methodology was selected to design the controller. Based on design requirements, an intuitive means of manipulating the LQR gains is presented. See [Ref. 9] for details. The following is a brief review of the properties of an LQR controller utilized in this design process.

A. LQR OVERVIEW

Consider the linear system

$$\mathcal{G} = \begin{cases} \dot{x} &= Ax + Bu \\ z &= C_1 x + D_1 u \\ y &= x \end{cases} \tag{3.1}$$

where $x \in R^n$, $u \in R^m$ and $z \in R^p$.

Suppose $C_1^T D_1 = 0$ and D_1 is full column rank. Then,

$$z^T z = x^T C_1^T C_1 x + u^T D_1^T D_1 u.$$

Define a cost, J, as follows:

$$J = \int (z^T z) dt = \int (x^T C_1^T C_1 x + u^T D_1^T D_1 u) dt \tag{3.2}$$

and let Q = $C_1^T C_1$, and R = $D_1^T D_1$. Note: Q $\geqq 0$ and R > 0.

Assume (C_1, A) is observable and (A, B) is controllable. Consider Figure 3.1. The standard LQR problem is to find a controller, $u = K(s)x$, such that the feed back system in Figure 3.1 is internally stable and J is minimized. It turns out, one such controller uses a constant gain, $u = -Kx$, where

$$K = -R^{-1}B^TP, \tag{3.3}$$

and P solves the Alegbraic Riccatti Equation:

$$A^TP + PA - PBR^{-1}B^TP + Q = 0. \tag{3.4}$$

Figure 3.1 shows the feedback interconnection of the plant $\mathcal{G}$ and the controller K. Here the inputs w can include commands and disturbances.

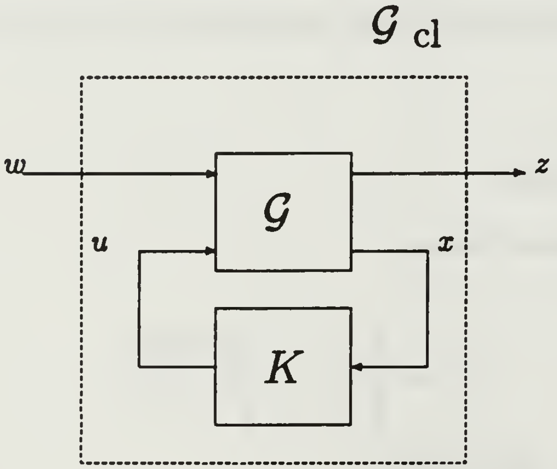


Figure 3.1: **Standard LQR feedback configuration.**

It turns out that the controller, K, has guaranteed simultaneous phase and gain margins of no less than 60 degrees and 3dB, respectively [Ref. 12]. Furthermore, the controller has asymptotic properties which are exploited in the design process and are discussed next.

Define the Hamiltonian matrix, H, as:

$$H = \begin{bmatrix} A & -BR^{-1}B^T \\ Q & -A^T \end{bmatrix}.$$

Let T be given by:

$$T = \begin{bmatrix} I & 0 \\ P & I \end{bmatrix}$$

then:

$$T^{-1} = \begin{bmatrix} I & 0 \\ -P & I \end{bmatrix}$$

where P solves the Riccati equation, Equation 3.4.

Note,

$$T^{-1}HT = \begin{bmatrix} A - BR^{-1}B^TP & -BR^{-1}B^TP \\ 0 & -A^T + PBR^{-1}B^T \end{bmatrix}.$$

Therefore the eigenvalues of H are the roots of the following polynomials:

$$det(sI - H) = det(sI - T^{-1}HT) = det(sI - A + BR^{-1}B^TP)det((sI + A^T - PBR^{-1}B^T)$$

Clearly, the eigenvalues of the Hamiltonian consist of the eigenvalues of $\mathcal{G}_{cl}$ and their unstable reflections about the imaginary axis. Let $R = \rho R_1,\ R_1 > 0$, then;

$$det(sI - H) = \begin{bmatrix} sI - A & (1/\rho)R_1^{-1}B^T \\ 0 & sI + A \end{bmatrix}.$$

It can be shown that the $det(sI - H)$ can be equivalently expresses as [Ref. 9]:

$$det(sI-H) = -1^n det(sI-A)det(-sI-A)det(I+(1/\rho)R_1^{-1}B^T(-sI-A^T)^{-1}C_1^TC_1(sI-A)^{-1}B)$$

Let

$$\phi(s) = det(sI - A)$$

and

$$\theta(s) = C_1(sI - A)^{-1}B.$$

Then

$$det(sI - H) = -1^n\phi(s)\phi(-s)det(I + (1/\rho)R_1^{-1}\theta(-s)\theta(s))). \qquad (3.5)$$

The SISO example will best demonstrate what occurs to the eigenvalues of $\mathcal{G}_{cl}$ as ρ is varied from 0 to ∞. Let

$$\theta(s) = \psi(s)/\phi(s)$$

where $\psi(s)$ are the zeros of $\theta(s)$. Then,

$$det(sI - H) = -1^n \phi(s)\phi(-s)(I + (1/\rho)\psi(s)\psi(-s)/\phi(s)\phi(-s)).$$

It follows that the eigenvalues of H are the roots of,

$$\phi(s)\phi(-s) + (1/\rho)\psi(s)\psi(-s). \tag{3.6}$$

Consider the feedback system shown in Figure 3.2. This is standard configuration for SISO root locus analysis and its characteristic equation is:

$$\phi(s)\phi(-s) + (1/\rho)\psi(s)\psi(-s),$$

exactly the same as Equation 3.6. Since we know that the stable eigenvalues of H are the eigenvalues of $\mathcal{G}_{cl}$, standard root locus techniques show that

as ρ goes to 0 [Ref. 9]:

- p eigenvalues of $\mathcal{G}_{cl}$ go to the stable zeros of $C_1(sI - A)^{-1}B$ or the stable reflection of the unstable zeros of $C_1(sI - A)^{-1}B$, where p is the number of zeros of $C_1(sI - A)^{-1}B$.

- n-p eigenvalues of $\mathcal{G}_{cl}$ go to -∞ in Butterworth patterns.

as ρ goes to ∞

- n eigenvalues of $\mathcal{G}_{cl}$ go to the stable eigenvalues of A and the stable reflections of the unstable eigenvalues of A.

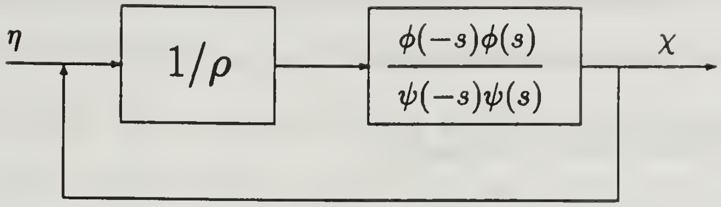


Figure 3.2: Feedback Configuration for Root Locus Analysis.

B. DESIGN REQUIREMENTS

These properties of an LQR controller were utilized in a design process in order to meet the following design requirements.

1. Zero Steady State Error

- Achieve zero steady state values for all error variables in response to ramp commands in position along the x, y, and z inertial axes. Note that a ramp command in postion corresponds to a constant heading, constant velocity trajectory.

2. Bandwidth Requirements

- The input-output command response bandwidth (command-loop bandwidth) along any of the three command channels should be no greater than 1 radian per second and no less than 1/10 radian per second.
- The control-loop bandwidth should not exceed 12 radians per second for the elevator and aileron actuators, and 5 radians per second for the throttle actuator. These numbers represent 80 % of the corresponding actuator

bandwidths and shall ensure the actuators are not driven beyond their linear operating range.

3. Closed Loop Damping

- The dominant closed loop eigenvalues should have a damping ratio of at least 0.7.

C. THE SYNTHESIS MODEL

The synthesis model is the primary interface between the control design and the LQR algorithm. At the heart of the synthesis model is a linear model of Bluebird developed in Chapter II.

Bluebird has four control inputs, namely elevator, rudder, ailerons, and throttle. The elevator and throttle are natural choices for controlling x and z position in steady state. The remaining two control inputs could be used to control the lateral variable (y position). Both rudder and aileron provide means of generating accelerations in the lateral plane. In fact, rudder is more effective at generating sideslip than aileron. In the linear plant, lateral position is the double integral of lateral acceleration. Subsequently, the resulting LQR controller will attempt to use rudder to null out errors in lateral position, i.e., to turn the plane. However, the desired controller response is to bank to turn using ailerons and to use rudder for turn coordination. Furthermore, in the presence of wind, it is desired that Bluebird fly wings level, crabbed into the wind, rather than use a wing down, top rudder technique. For these reasons, the rudder was removed as a control input to the linear model.

As can be seen from Table 3.1, the dutch roll mode of Bluebird is lightly damped. This light damping of 0.111 could pose a performance problem. Since rudder is available and not used in the design of the trajectory controller, the nonlinear model of Bluebird was modified to include a yaw damper for improved dutch roll damping. Yaw rate was fed back to the rudder through a constant gain block with a value of 0.55. Additionally, the rudder was removed as an external input. Note that Bluebird is still fully controllable with the remaining three control inputs.

TABLE 3.1: EIGENVALUES OF BLUEBIRD

Mode	Frequency	Damping
Longitudinal	*rad/sec*	
Short Period	5.9	0.735
Phugoid	0.497	0.0344
Lateral–Directional		
Dutch Roll	2.4	0.111
Spiral	0.0384	-1
Roll Response	-4.572	1

The nonlinear model of Bluebird with three inputs and integral yaw damper was linearized, as per Chapter II, returning the linear model,

$$\mathcal{G} = \begin{cases} \dot{x} & = & Ax + Bu \\ y & = & x \end{cases} \tag{3.7}$$

where

$$x = \begin{bmatrix} u & v & w & p & q & r & \Phi & \Theta & \Psi & Xpos & Ypos & Zpos & \delta_{e_{pos}} & \delta_{a_{pos}} & \delta_{t_{pos}} \end{bmatrix}^T$$

and

$$u = \begin{bmatrix} \delta_{elevator} & \delta_{aileron} & \delta_{throttle} \end{bmatrix}^T.$$

This linear model was used in the LQR design. The eigenvalues of Bluebird with a yaw damper are given in Table 3.2 where it can be seen that the dutch roll mode has been damped out. Note that the number of states decreases to fifteen since the rudder actuator state was removed.

TABLE 3.2: EIGENVALUES OF BLUEBIRD WITH YAW DAMPER

Mode	Frequency	Damping
Longitudinal	*rad/sec*	
Short Period	5.9	0.735
Phugoid	0.497	0.0344
Lateral–Directional		
Dutch Roll	2.35	0.5
Spiral	0.1788	1
Roll Response	-4.5686	1

Consider Figure 3.3. Here K is the controller to be designed, G is the linear model of Bluebird, and the block, S, within the dotted line is the synthesis model.

The signal, w, represents the commanded trajectory inputs:

$$w = \begin{bmatrix} Xpos_{cmd} & Ypos_{cmd} & Zpos_{cmd} & \Phi_{cmd} & \Theta_{cmd} & \Psi_{cmd} \end{bmatrix}^T .$$

The signal x_1 represents the linear and angular position states in the linear model.

$$x_1 = \begin{bmatrix} Xpos & Ypos & Zpos & \Phi & \Theta & \Psi \end{bmatrix}^T$$

The signal e represents the errors between the commanded and current trajectory. The signal z is comprised of the outputs of the matrices, Q, and R. Since zero steady state error is desired while tracking a ramp command in inertial position, two integrators were placed on each error signal. This also ensures

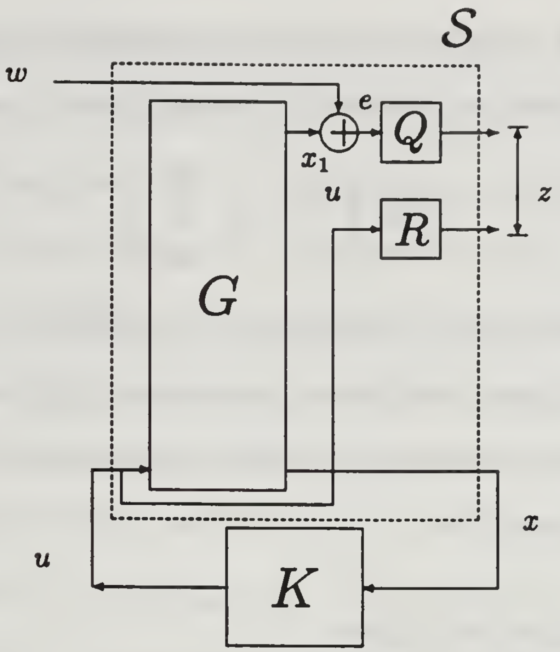


Figure 3.3: Synthesis and Analysis Model

perfect tracking of constant heading trajectories in the presence of a constant wind disturbance. Thus Q was chosen to be;

$$Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \begin{bmatrix} \frac{c_{11}}{1} & \frac{c_{12}}{s} & \frac{c_{13}}{s^2} \\ \frac{c_{21}}{1} & \frac{c_{22}}{s} & \frac{c_{23}}{s^2} \\ \frac{c_{31}}{1} & \frac{c_{32}}{s} & \frac{c_{33}}{s^2} \end{bmatrix}$$

The values of c_{xx} were chosen to place six transmission zeros from u to z at appropriate locations. If they are well chosen, six poles in the closed loop plant will move very near to the placed transmisssion zeros. With that in mind, the transmission zeros were chosen as appropriate target locations for the poles added by the addition of the error states.

The q_{xx} weightings are used as a mechanism for obtaining the desired command bandwidth. Increasing the value of q_{xx} increases the relative proportion of that error state in the regulated output vector z. The resultant LQR gain increases the command bandwidth in that channel in order to move the controlled state to its commanded value more quickly.

The matrix R is a constant diagonal matrix required to be full rank. Since the plant $\mathcal{G}$ has three control inputs, R is of the form,

$$R = \begin{pmatrix} r_{11} & 0 & 0 \\ 0 & r_{22} & 0 \\ 0 & 0 & r_{33} \end{pmatrix}$$

The elements of R are used as a mechanism for selecting the control bandwidth. An increase in r_{xx} increases the relative proportion of that actuator energy in the regulated output z. The resultant LQR gain decreases the control bandwidth of that control input.

D. THE DESIGN PROCESS

Design requirements given in the previous section are SISO in nature. They are expressed as bandwidth limitations of the individual actuators and rise time and damping characteristics along the command channels. Note, the rise time is inversely proportional to the command bandwidth. The following LQR design process provided a means of obtaining a multivariable solution to achieve SISO design specifications.

With an appropriate linear representation of Bluebird and a synthesis model that incorporated well placed transmission zeros, the design "knobs" were adjusted in order to meet performance requirements. The design "knobs" are the elemental weightings, q_{xx} and r_{xx}, in the Q and R matrices. The design process iterated through the following steps.

1. Initially let q_{xx} equal 1. Iteratively determine weights for R to satisfy control loop bandwidth requirements. Increasing r_{xx} decreases the control bandwidth along that channel.

2. With R from step 1, iteratively determine weights for q_{xx} to satisfy

command loop bandwidth requirements. Increasing q_{xx} increases the command bandwidth along that channel and decreases the rise time.

3. If it is required to increase the damping of a lightly damped mode, use an eigenvector decomposition to determine the primary states affecting that mode. Include a weighting on the derivative of those states in the output z.

4. Ensure that control-loop bandwidths are still satisfactory with the values of q_{xx} in step 2. It is possible that all of the performance requirements are not acheivable within control bandwidth limitations.

5. Connect the LQR controller to the linear plant and evaluate the performance in terms of command response and disturbance rejection.

6. Confirm satisfaction of other design requirements, including damping.

7. If any step is unsatisfactory, go back to the synthesis model and make appropriate changes. Transmission zeros may need to be added, moved, or deleted. Synthesis model outputs may need to be reevaluated.

Bode plots were used to determine compliance with the requirements. After five iterations through the seven step process, the following values for Q and R matrices resulted in a controller design that met design requirements.

$$Q = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{0.2}{s} & \frac{0.01}{s^2} \\ 1 & \frac{0.4}{s} & \frac{0.0625}{s^2} \\ 1 & \frac{0.4}{s} & \frac{0.0625}{s^2} \end{bmatrix}$$

$$R = \begin{bmatrix} 5000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix}$$

The transmission zeros created in the sythesis model are shown in Table 3.3.

TABLE 3.3: TRANSMISSION ZEROS OF SYNTHESIS MODEL

Channel	c_{x1}	c_{x2}	c_{x3}	Freqency (rad/sec)	Damping
X_{pos}	1	0.2	0.01	0.1	1
Y_{pos}	1	0.4	0.0625	0.25	0.8
Z_{pos}	1	0.4	0.0625	0.25	0.8

E. LQR CONTROLLER PERFORMANCE

The eigenvalues of the feedback interconnection of the plant and controller are given in Table 3.4. It is apparent that the zeros created in the synthesis model were well placed and attracted the integrators created by the addition of the error states. There are two sets of lightly damped poles. These do not present a problem because their frequency is an order of magnitude greater than the frequency of the eigenvalues associated with the trajectory commands. Notice that the actuator poles did not change, indicating that the control bandwidths were slower than the actuator bandwidths. Actuator models provide a simple means of determining if the control bandwidths exceeded the actuator bandwidths.

Figures 3.4 through 3.6 depict the control-loop bandwidths for the elevator, aileron, and throttle. The cross coupling between longitudinal and lateral flight controls was so slight that it is not shown due to scale.

Figures 3.7 through 3.9 depict the command-loop bandwidth for step commands in inertial position. Notice that there is some coupling between X command and Z response; the rest are essentially uncoupled.

A summary of the resulting command and control bandwidths achieved is presented in Table 3.5.

TABLE 3.4: EIGENVALUES OF THE FEEDBACK SYSTEM

Mode	Frequency (rad/sec)	Damping
X Axis Response	0.08	0.92
Y Axis Response	0.21	0.77
Z Axis Response	0.21	0.77
elevator	12.1	1.0
aileron	12.3	1.0
throttle	12.4	1.0
others	0.62	0.79
	1.90	0.35
	2.17	1.0
	2.35	0.18
	2.26	0.68
	4.54	1.0
	5.86	0.74

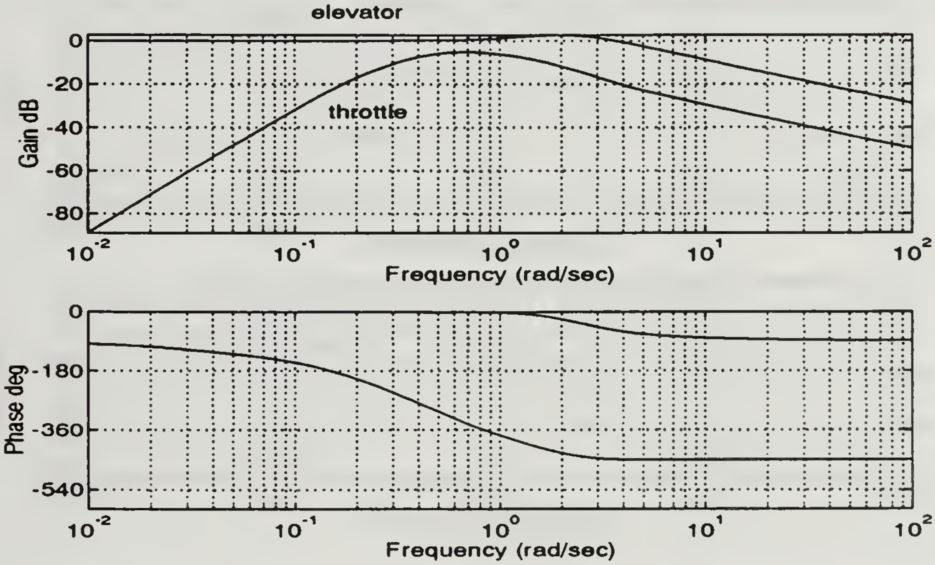


Figure 3.4: Control-Loop Bandwidth: Elevator Channel

The response to two types of trajectory commands is of interest. The first is the reponse to a ramp command in Y position. This corresponds to a change in the heading of the commanded trajectory. The response in terms of angle of bank and heading activity is shown in Figure 3.10. The controller achieves the

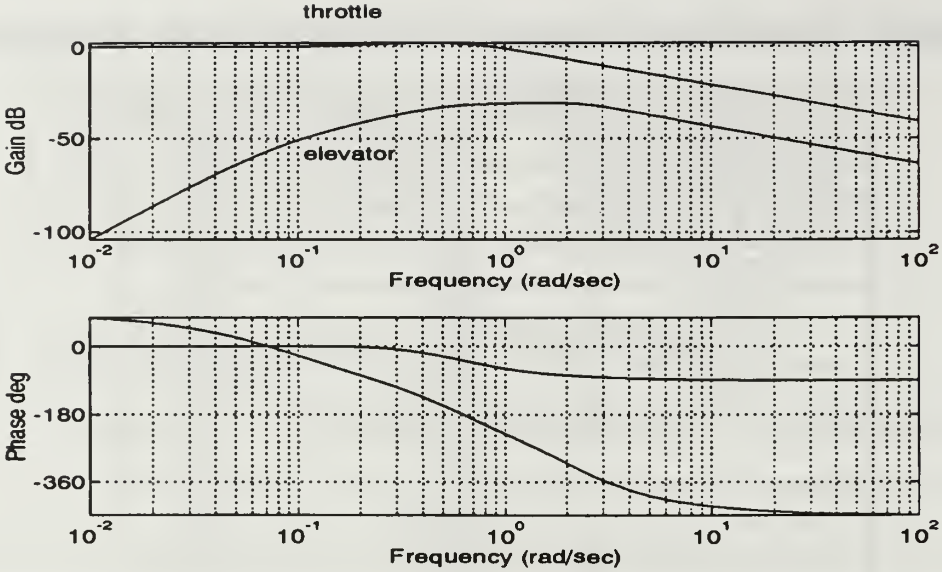


Figure 3.5: Control-Loop Bandwidth: Throttle Channel

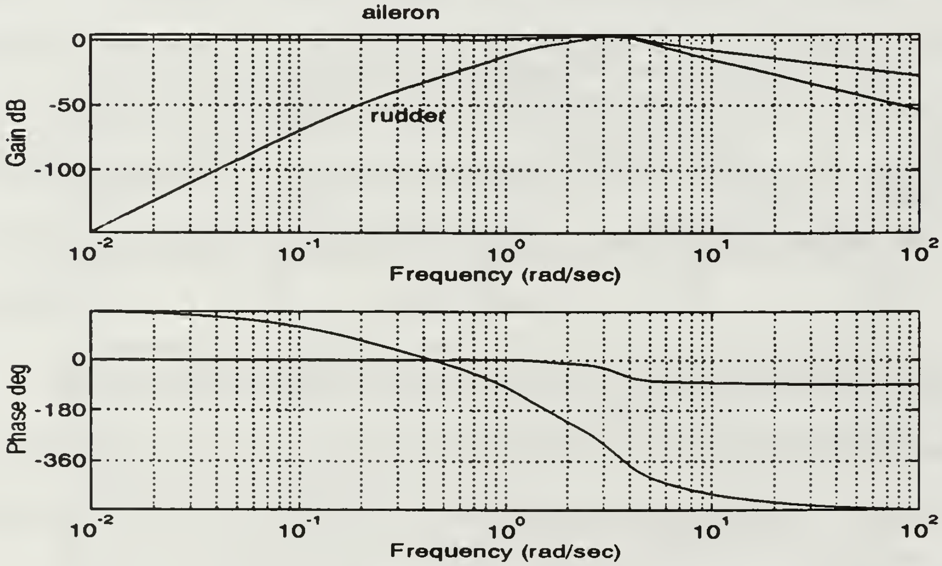


Figure 3.6: Control-Loop Bandwidth: Aileron Channel

desired result of turning Bluebird to the required heading. The nonminimum phase response of the heading state is due to adverse yaw.

The response to a ramp command in Z position corresponds to the response to a change in flight path angle of the commanded trajectory. Figure 3.11

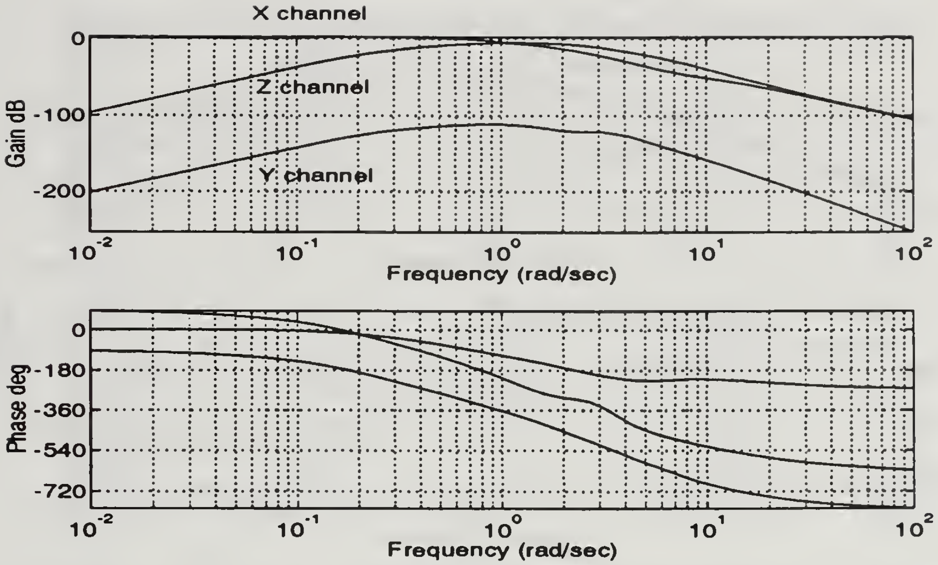


Figure 3.7: Command-Loop Bandwidth: X Position Channel

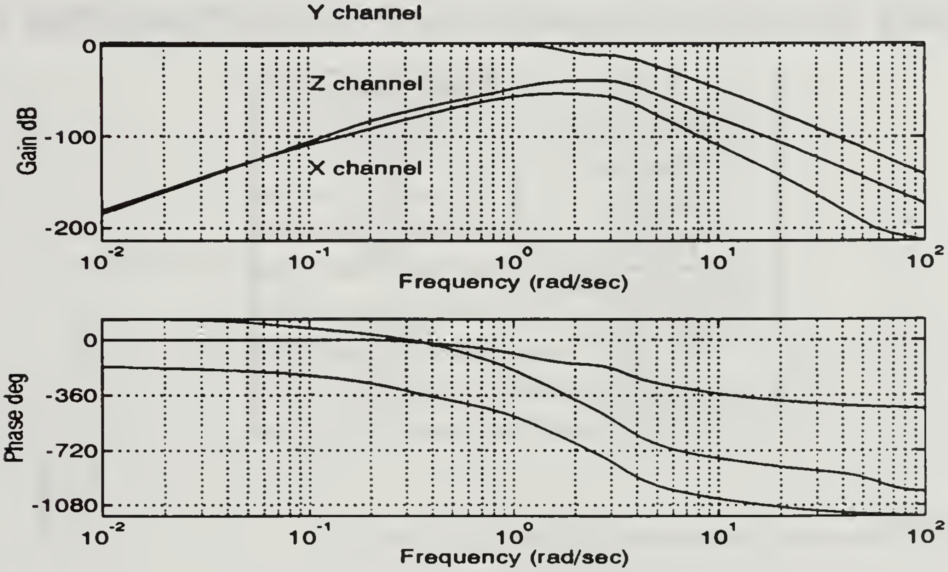


Figure 3.8: Command-Loop Bandwidth: Y Position Channel

shows the response to this command in terms of pitch angle, θ, and altitude z. Note that the positive z direction is down, hence the negative pitch angle.

Finally, the response to a constant wind disturbance is shown in Figure 3.12. The wind orientation is such that it affects all three axes of Bluebird

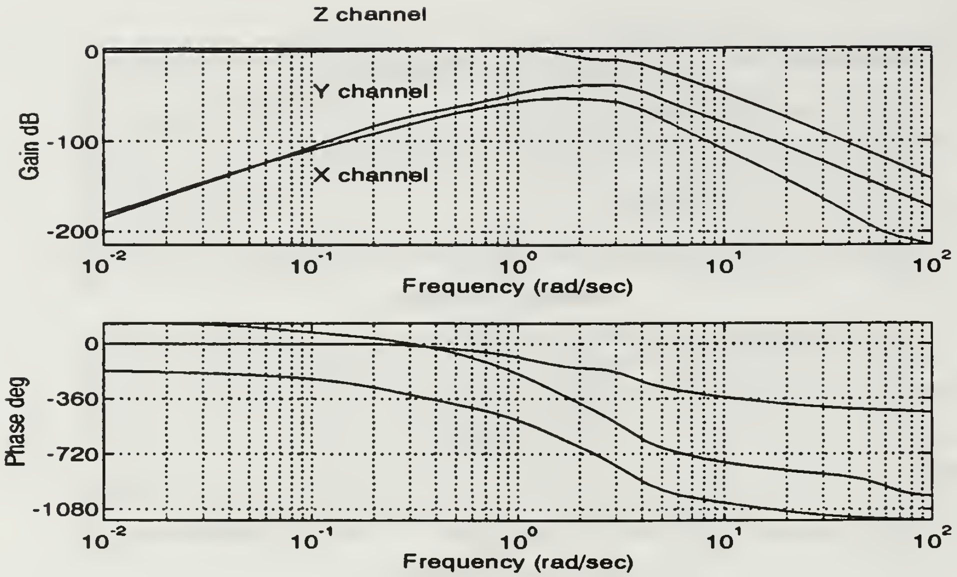


Figure 3.9: Command-Loop Bandwidth: Z Position Channel

TABLE 3.5: COMMAND AND CONTROL BANDWIDTHS

Loop	Break Frequency
Control Loop	*rad/sec*
Elevator	4.0
Aileron	5.0
Throttle	1.0
Command Loop	
X Axis	0.75
Y Axis	1.0
Z Axis	1.0

equally. The wind disturbance begins at t = 0.

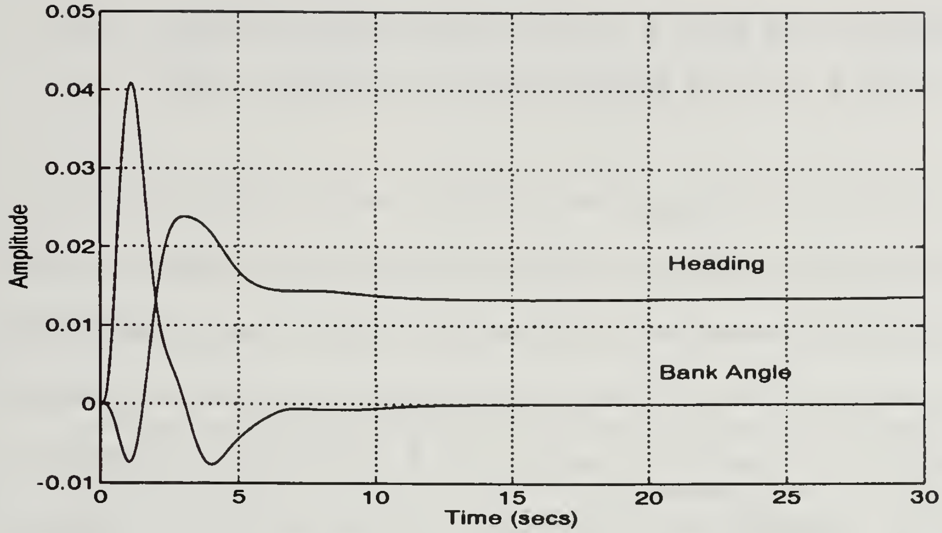


Figure 3.10: Bank Angle and Heading Response to Ramp in Y Command

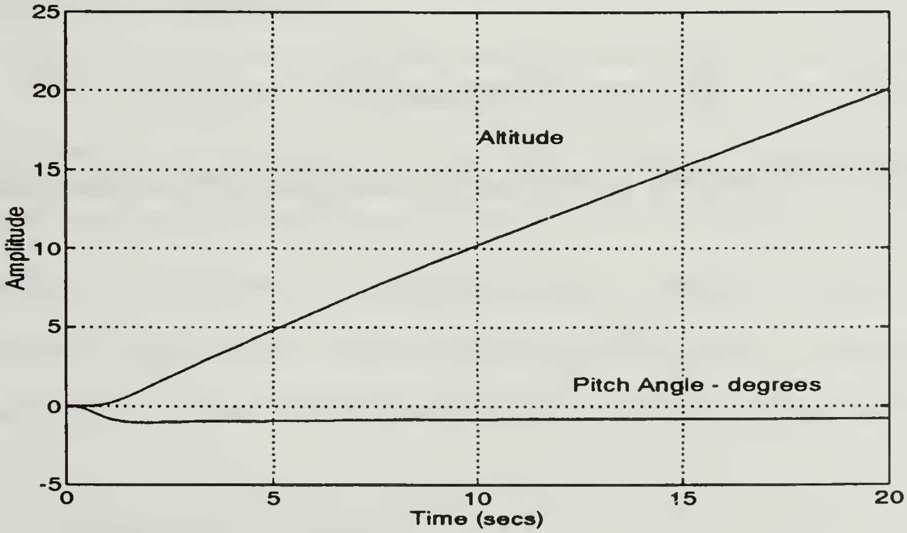


Figure 3.11: Pitch Angle and Altitude Response to Ramp in Z Command

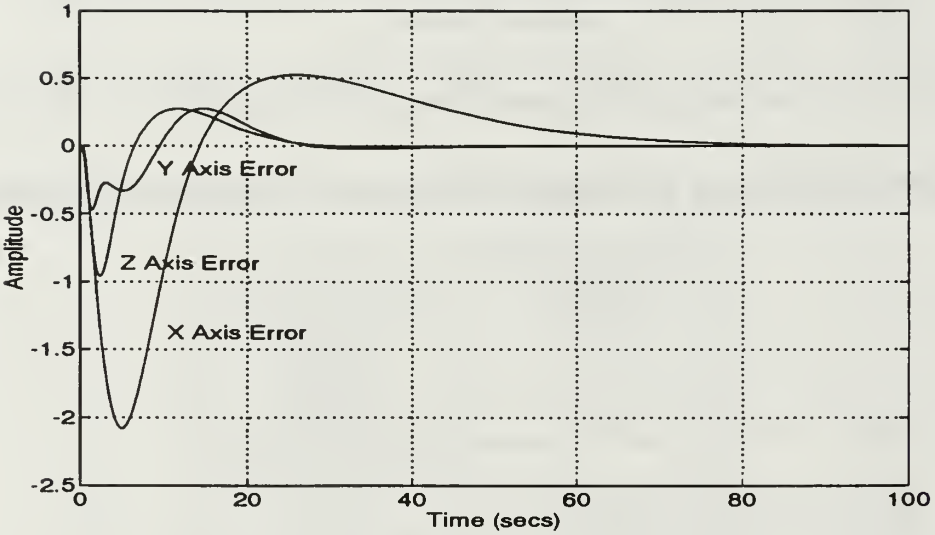


Figure 3.12: Trajectory Error Due to a Constant Wind Disturbance

IV. CONTROLLER IMPLEMENTATION ON THE NONLINEAR PLANT

In Chapter III, a linear controller was designed to control the trajectory of an air vehicle. However, this controller cannot be implemented, as is, on the nonlinear plant. It would only be effective for commanded trajectories that represent relatively small perturbations from the specific trajectory for which it was designed. Intuitively, it is clear that the dynamics of the plant do not depend on the heading angle Ψ. Furthermore, the orientation of the force of gravity is the only change in the dynamics of the air vehicle due to changes in Φ or Θ. It turns out that these issues were addressed in [Ref. 10], where a new methodology for implementing controllers on nonlinear plants is proposed. The method involves differentiating some of the inputs to the controller, hence the term, $\mathcal{D}$-implementation.

This chapter begins with a general description of the structure of $\mathcal{D}$-implementation. Furthermore, the specifics of its implementation on the nonlinear model in SIMULINK are discussed. Next, the fidelity of the nonlinear simulation is improved by incorporating output feedback. This step involves inclusion of high fidelty sensor models and Kalman filters. Finally, all of the pieces of the complete nonlinear simulation are brought together in SIMULINK.

A. $\mathcal{D}$-IMPLEMENTATION

Using the development in Chapter II, the vehicle dynamics can be expressed in state space form as follows:

$$\begin{aligned}
\frac{d}{dt}\,{}^{B}v_{cg} &= f_v({}^{B}v_{cg}, {}^{B}\omega_B, u) + {}_{I}^{B}R(\Lambda)g \\
\frac{d}{dt}\,{}^{B}\omega_B &= f_\omega({}^{B}v_{cg}, {}^{B}\omega_B, u) \\
\frac{d}{dt}\,P_{cg} &= {}_{B}^{I}R(\Lambda)\,{}^{B}v_{cg} \\
\frac{d}{dt}\,\Lambda &= S(\Lambda)\,{}^{B}\omega_B,
\end{aligned} \tag{4.1}$$

where f_v and f_ω are continuously differentiable and $u \in \mathcal{R}^6$ denotes the vector of control inputs. To condense the notation, we define

$$x_v := \begin{bmatrix} {}^{B}v_{cg} \\ {}^{B}\omega_B \end{bmatrix}, \quad x_p := \begin{bmatrix} P_{cg} \\ \Lambda \end{bmatrix}, \quad f_1(.) := \begin{bmatrix} f_v(.) \\ f_\omega(.) \end{bmatrix},$$

$$f_2(.) := \begin{bmatrix} {}_{I}^{B}R(.)g \\ 0 \end{bmatrix}, \quad L(.) := \begin{bmatrix} R(.) & 0 \\ 0 & S(.) \end{bmatrix},$$

where $x_1 \in R^6$, $x_2 \in R^6$ and $L \in R^{6x6}$. Furthermore, we define

$$r := \begin{bmatrix} P_c \\ \Lambda_c \end{bmatrix} \in R^6, \tag{4.2}$$

as the vector of linear and angular position commands that Bluebird must track. With this notation, the dynamics of the augmented plant can be written as follows:

$$\mathcal{G} := \begin{cases} \dot{x}_v &= f_1(x_v, u) + f_2(x_p) \\ \dot{x}_p &= L(x_p)x_v \\ e_1 &= [I \;\; 0](y_2 - r) \\ e_2 &= [0 \;\; I](y_2 - r) \\ y_1 &= h(x_v, u) \\ y_2 &= x_p \end{cases} \tag{4.3}$$

where y_1 and y_2 are the available measurements, e_1 and e_2 are the trajectory errors separated into linear and angular components. Notice that L is only a function of the orientation vector $\Lambda = [0 \;\; I]x_p$. For simplicity of exposition, we have not included any extra dynamics for the actuators or sensors.

The set $\mathcal{E}$ of trajectories where the plant $\mathcal{G}$ is expected to operate is given by

$$\mathcal{E} \;:=\; \{(x_{v_0}, x_{p_0}, u_0, r_0) : \begin{bmatrix} x_{v_0} = \begin{bmatrix} v_0 \\ \omega_0 \end{bmatrix} \\ x_{p_0} = r_0 \\ f_1(x_{v_0}, x_{p_0}, u_0) + f_2(x_{p_0}) = 0 \end{bmatrix}, r_0 \in \Gamma\},$$

where Γ corresponds to the set of prescribed linear and angular positions of Bluebird. This is a broad definition of trimmed flight. Notice that it does not preclude the presence of an inertial acceleration due to centripetal force. As usual, we restrict the angular positions to some subset of $[-\pi/2, \pi/2] \times (-\pi/2, \pi/2) \times [-\pi/2, \pi/2]$, as the inverse of $L(.)$ is not defined at $\theta = \pi/2$. Notice, from the definition of $x_{v_0} \in \mathcal{E}$ and equation (4.1) it follows that:

$$^{B}\omega_B \in \mathcal{E} \rightarrow^{B} \omega_B = constant$$

$$\Lambda \in \mathcal{E} \rightarrow \dot{\Lambda} = constant.$$

Notice that the set $\mathcal{E}$ is easily parameterized by $x_{p_0} = r_0 \in \Gamma$. Given $(x_{v_0}, x_{p_0}, u_0, r_0) \in \mathcal{E}$, we obtain

$$y_{1_0} := h(x_{v_0}, u_0)$$

$$y_{2_0} := x_{p_0}$$

$$e_{1_0} := [I \;\; 0](y_{2_0} - r_0) := 0$$

$$e_{2_0} := [0 \;\; I](y_{2_0} - r_0) := 0..$$

Let δx_v, δx_p, $\delta u, \delta y_1$, δy_2, δr, δe_1 and δe_2 correspond to small perturbations of x_v, x_p, u, y_1, y_2, r, e_1, and e_2 about the nominal values x_{v_0}, x_{p_0}, u_0, y_{1_0}, y_{2_0}, r_0, e_{1_0}, and e_{2_0} respectively. The family of linearized models associated with the rigid body $\mathcal{G}$ and the set $\mathcal{E}$ is defined as

$\mathcal{G}_l := \{\mathcal{G}_l\,(r_0),\ r_0 \in \Gamma\}$, where

$$\mathcal{G}_l\,(r_0) := \begin{cases} \dot{\delta x}_v &= A_1\delta x_v + A_2\delta x_p + B\delta u \\ \dot{\delta x}_p &= L\delta x_v + A_4\delta x_p \\ \delta y_1 &= C_1\delta x_v + D_1\delta u \\ \delta y_2 &= \delta x_p \\ \delta e_1 &= [I\ \ 0](\delta y_2 - \delta r) \\ \delta e_2 &= [0\ \ I](\delta y_2 - \delta r) \\ \alpha_0 &= v(x_{v0}, r_0, u_0), \end{cases} \tag{4.4}$$

and

$$A_2 = \begin{bmatrix} 0 & g \times^B_I R(\Lambda_0) S^{-1}(\Lambda_0) \\ 0 & 0 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 0 & -^I_B R(\Lambda_0) V_0 \times S^{-1}(\Lambda_0) \\ 0 & 0 \end{bmatrix}. \tag{4.5}$$

Matrices A_2 and A_4 were derived using the identity in Appendix C. The intent of this derivation is to isolate the plant dynamics that are a function of Λ_0. Notice that A_2 represents the contribution of the force of gravity to the dynamics of Bluebird and A_4 represents the sensitivity of Bluebird's trajectory to changes in the air vehicle's spatial orientation.

Let $r_0 \in \mathcal{E}$ be given. Define

$$\mathcal{G}_{l_0}\,(r_0) = \begin{cases} \dot{\delta x}_v &= A_1\delta x_v + \bar{A}_2\delta x_p + B\delta u \\ \dot{\delta x}_p &= \delta x_v + \bar{A}_4\delta x_p \\ \delta y_1 &= C_1\delta x_v + D_1\delta u \\ \delta y_2 &= \delta x_p \\ \delta e_1 &= [I\ \ 0](\delta y_2 - \delta r) \\ \delta e_2 &= [0\ \ I](\delta y_2 - \delta r) \\ \alpha_0 &= v(x_{v0}, r_0, u_0), \end{cases} \tag{4.6}$$

where

$$\bar{A}_2 = \begin{bmatrix} 0 & g \times^B_I R(\Lambda_0) \\ 0 & 0 \end{bmatrix}, \quad \bar{A}_4 = \begin{bmatrix} 0 & -v_0 \times \\ 0 & 0 \end{bmatrix}. \tag{4.7}$$

Notice that Equation 4.6 decribes the linear model of Bluebird used for the design of the LQR controller in the previous chapter, where r_0 was chosen as the origin of $\{I\}$ with Λ_0 equal to zero. Note, at this condition, ${}^B_I R(0) = I$. Recall, the structure of the controller developed in Chapter III is given by:

$$\mathcal{K}_l = \begin{cases} \delta\dot{x}_{c1} &= B_{c2}\delta x_{c2} \\ \delta\dot{x}_{c2} &= [\delta e_1 \;\; 0]^T \\ \delta u &= C_{c1}\delta x_{c1} + C_{c2}\delta x_{c2} + D_{c1}\delta y_1 + [D_{c3} \;\; D_{c2}][\delta e_1 \;\; \delta e_2]^T. \end{cases} \tag{4.8}$$

Based on $\mathcal{K}_l$, we propose to implement the following controller for the nonlinear plant $\mathcal{G}$:

$$\mathcal{K}\,(\Lambda) := \begin{cases} \dot{x}_{c1} &= B_{c2}L^{-1}(\Lambda)[e_1 \;\; 0]^T \\ \dot{x}_{c2} &= C_{c1}x_{c1} + C_{c2}L^{-1}(\Lambda)[e_1 \;\; 0]^T + D_{c1}\dot{y}_1 \\ &\quad + D_{c2}L^{-1}(\Lambda)[0 \;\; \dot{e}_2]^T \\ \Lambda &= [0 \; I]y_2 \\ u &= x_{c2} + D_{c3}L^{-1}(\Lambda)[e_1 \;\; 0]^T. \end{cases} \tag{4.9}$$

Figure 4.1 shows the block diagram of $\mathcal{K}\,(\Lambda)$. Notice that Λ serves as a gain scheduling variable. As a function of Λ, L^{-1} serves to properly resolve the trajectory error at the input to the controller. Note, the controller forms the derivative of the measured outputs, y_1. Recall, y_1 is the measurement of the states x_v and the dynamics of x_v are essentially independent of the air vehicle's spatial orientation or position in $\{I\}$. An integrator at the output of the controller serves to recover the properties of the linear design. The error is formed using the outputs y_2 and the commanded trajectory r. Recall, y_2 is the measurement of the states x_p. L^{-1} serves to resolve this error, originally formed in $\{I\}$, in $\{B\}$.

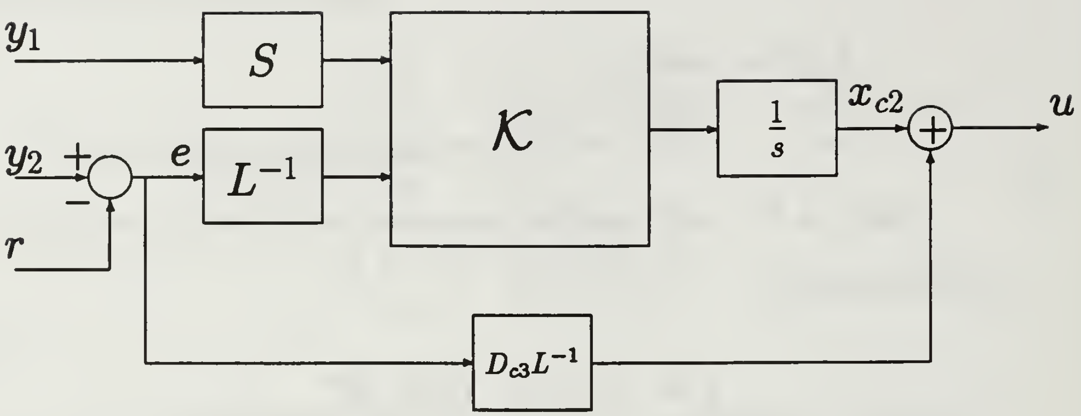
It turns out that the implementation of Figure 4.1 has an important property discussed next. First we need to make the following assumptions:

*A*1. $Dim(x_{c2}) = dim(u) = dim(y_2)$.

*A*2. The matrix

$$\begin{bmatrix} sI - A_{c1} & B_{c2} \\ -C_{c1} & C_{c2} \end{bmatrix}$$

has full rank at $s = 0$.



$$\mathcal{K} = \left[\begin{array}{cc|cc} \multicolumn{2}{c|}{0} & 0 & B_{c2} \\ \hline C_{c1} & C_{c2} & D_{c1} & [D_{c3}\ D_{c2}] \end{array}\right]$$

Figure 4.1: Block diagram of the nonlinear controller $\mathcal{K}\ (\Lambda)$

*A*3. The matrix pair (A_{c1}, C_{c1}) is observable.

Also denote by $\mathcal{T}(\mathcal{G}_{l_0}\ \ \mathcal{K}_l\ ) : \delta r \rightarrow \delta y_2$ the closed loop linear system that results from connecting $\mathcal{G}_{l_0}$ to $\mathcal{K}_l$, and by $T(\mathcal{G}_{l_0}\ \ \mathcal{K}_l\ )(s)$ its transfer matrix. Similarly, we let $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{K}\ )(r_0)$ denote the linearization of the closed loop system $\mathcal{T}(\mathcal{G}\ ,\ \mathcal{K}\ )$ at the equilibrium point determined by r_0 and let $T_l(\mathcal{G}\ ,\ \mathcal{K}\ )(r_0)(s)$ be its transfer matrix. Then the following hold.

- the feedback systems $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{K}\ )(r_0)$ and $\mathcal{T}(\mathcal{G}_{l_0}\ ,\ \mathcal{K}_l\ )$ have the same closed loop eigenvalues;
-

$$\begin{aligned} T_l(\mathcal{G}\ ,\mathcal{K}\ )(r_0)(s) = \quad & L(\Lambda_0)T(\mathcal{G}_{l_0}\ ,\ \mathcal{K}_l\ )(s)L^{-1}(\Lambda_0), \\ & \Lambda_0 = [0\ I]x_{p0} \end{aligned}$$

Thus, the eigenvalues of the linearizations at each operating point are preserved; furthermore, the input-output behavior of the linearized operators is preserved in a well-defined sense. The reader is referred to [Ref. 10] for a complete discussion on approximations to this method that avoid using pure differentiation. The proof of this result is contained in Appendix C.

B. $\mathcal{D}$-IMPLEMENTATION OF THE CONTROLLER IN SIMULINK

In general, three steps are required to implement the linear controller developed Chapter III on the nonlinear plant. First, the controller needs to form the derivative of x_v as per Equation 4.9. Recall, $\dot{x_v}$ is equal to the vector $[{}^B\dot{v}_{cg}, {}^B\dot{\omega}_B]$. The first three elements of the vector, ${}^B\dot{v}_{cg}$, are available as processed acceleration outputs from the IMU. Therefore, the controller need not compute the derivative of ${}^Bv_{cg}$ since it will have it available directly. The derivative of ${}^B\omega_B$ is computed by the controller.

Secondly, the controller needs to act on the vectors, e_1 and e_2. The linear position error e_1 is formed as the difference in commanded and current position. The vector e_1 is then multiplied by the transformation matrix ${}^B_IR$. This effectively resolves the linear trajectory error in the body-fixed reference frame. Along this position trajectory, there exists a corresponding trajectory of Euler angles. Recall that $x_v =$ constant is one of the constraints on the set of trajectories. This includes a broad range of flight conditions such as steady turns, steady pull-ups, climbing or descending turns, or constant heading. While it is natural to define the linear trajectory as a sequence of positions, it is more convenient to define the derivatives of the Euler angles rather than the values of the angles directly. Consider, for example, that for many trajectories of interest in $\mathcal{E}$, the components of $\dot{\Lambda}$ can be described by: $\dot{\Phi} = 0$; $\dot{\Theta} = 0$; and $\dot{\Psi} =$ desired turn rate. Furthermore, the relationship between the rates of change of the Euler angles and ${}^B\omega_B$ is used as a means of resolving the angular position error in $\{B\}$. Recall,

$$^{B}\omega_{B} = S^{-1}(\Lambda)\dot{\Lambda}. \tag{4.10}$$

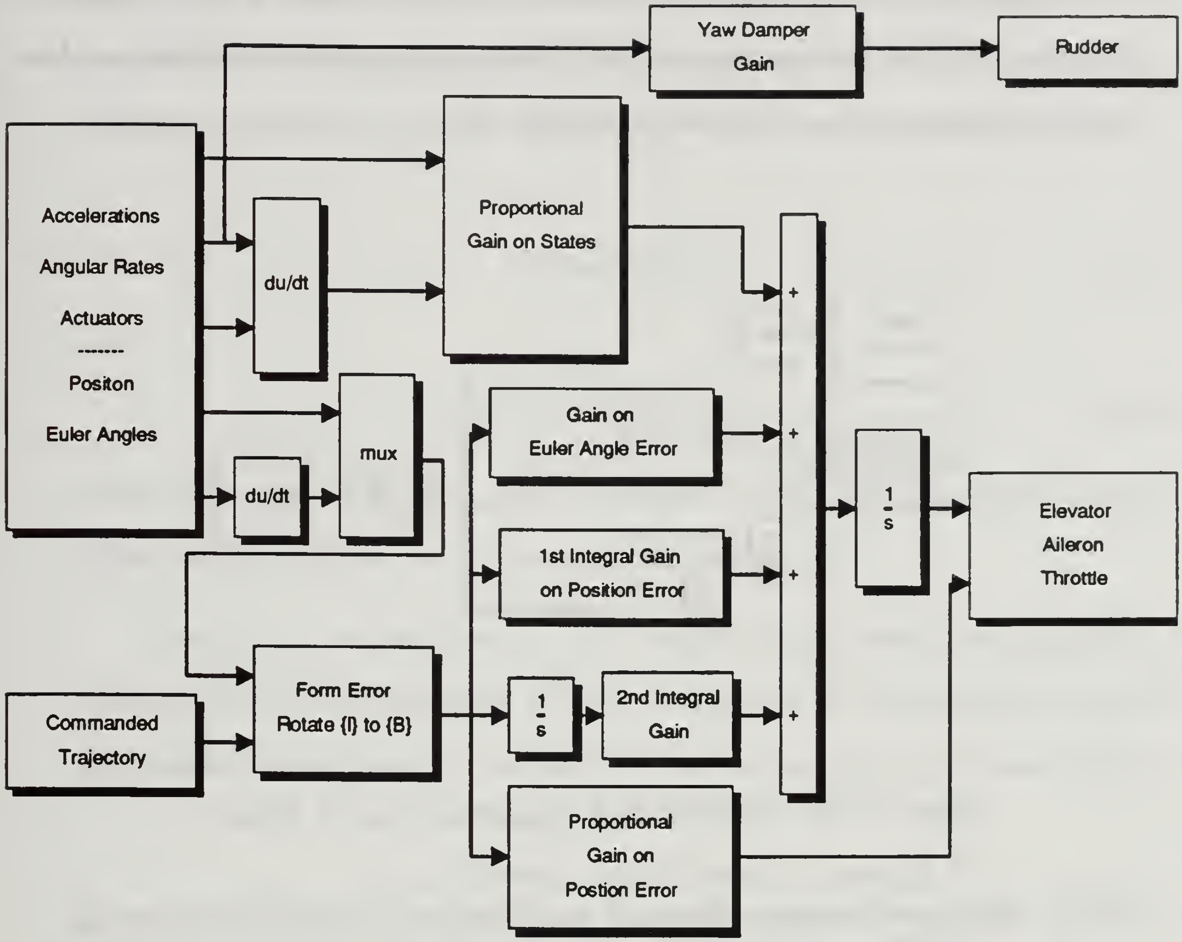
Therefore, the derivative of the Euler angle states is formed and the commanded Euler angle rate is removed to form $\dot{e_2}$. This error is resolved in $\{B\}$ using Equation 4.10. The integrator at the end of the controller recovers the effect of the initial differentiation.

Thirdly, the required error states are formed by integrating the rotated linear trajectory error vector e_1. Figure 4.1 indicates that integral action is accomplished at the output of the controller in order to recover the original properties of the linear design. This accounts for one of the integration steps on the error. Therefore, only one additional integrator is required to provide double integration action on the trajectory error e_1.

Figure 4.2 shows the $\mathcal{D}$-implemented controller in SIMULINK, file *plant1.m*. See Appendix B for a complete description. The LQR gain has been parsed into several separate matrices for clarity of control action.

C. GENERATION OF THE TRAJECTORY COMMANDS

The commanded trajectory is specified with respect to the inertial reference frame. At this point, it is assumed that a knowledge of the air vehicle's performance capabilities is known and that the specified trajectory is within those capabilities provided there is no wind. The air vehicle has certain airspeed restrictions with respect to the air mass that cannot be violated regardless of the desired trajectory to be tracked. These restrictions typically provide lower and upper bounds on the velocity of the air vehicle with respect to the airmass.


Yaw Damper Gain
Rudder
Accelerations
Angular Rates
Actuators

Positon
Euler Angles
du/dt
Proportional Gain on States
mux
du/dt
Gain on Euler Angle Error
1st Integral Gain on Position Error
1/s
Elevator
Aileron
Throttle
Form Error Rotate {I} to {B}
Commanded Trajectory
1/s
2nd Integral Gain
Proportional Gain on Postion Error
+
+
+
+

Figure 4.2: $\mathcal{D}$-Implementation of controller on Bluebird

An example of a lower limit is the stall speed of the air vehicle. Such limits are usually based on fundamental physical limitations of the airframe and a "flyable" trajectory can become "unflyable" under certain conditions. Therefore, a logic block positioned between the commanded trajectory and the controller ensures that the commanded trajectory can be flown at current flight conditions within user defined *indicated* airspeed limits, shown in Figure 4.3. The commanded linear trajectory enters the block as a time stamped position fix in the inertial reference frame. Onboard sensors provide both inertial velocities from the IMU and air mass velocities from the pitot-static system. Furthermore, a dual vane device instrumented on Bluebird provides both α and β measurements. Note: a close approximation to these readings could be obtained from IMU measurements as outlined in Chapter II.

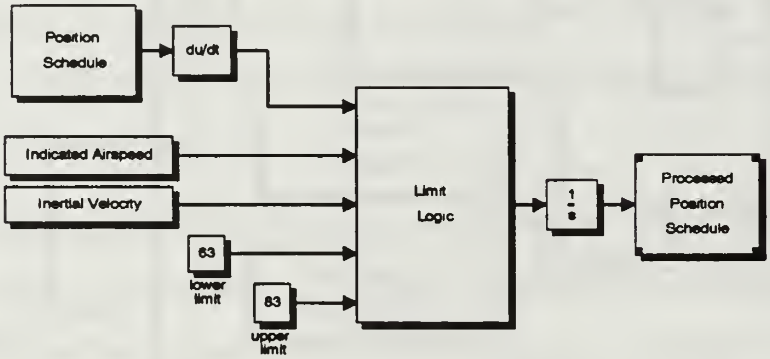

Position Schedule
du/dt
Indicated Airspeed
Inertial Velocity
63
lower limit
83
upper limit
Limit Logic
1/s
Processed Position Schedule

Figure 4.3: Commanded Trajectory Logic Block

With these measurements, the wind vector resolved in $\{I\}$ is calculated as:

$$ {}^IW = {}^I_BR\,{}^Bv_{cg} - {}^I_BR^B_WR\,{}^Wv_{cg} \tag{4.11}$$

where

$$ {}^Wv_{cg} = \begin{bmatrix} V_t \\ 0 \\ 0 \end{bmatrix} $$

and V_t is the indicated airspeed obtained from the pitot-static system.

The commanded indicated airspeed of the air vehicle, V_{t-cmd}, is calculated as:

$$ V_{t-cmd} = |{}^Iv_{cmd} - {}^IW|, $$

where ${}^{I}v_{cmd}$ is the numeric derivative of the commanded linear trajectory. If the commanded indicated airspeed is not within specified limits, the commanded trajectory is altered as follows. The angles θ and ψ are calculated as follows:

$$\theta = tan^{-1}(v_z/v_x) \tag{4.12}$$

and

$$\psi = sin^{-1}(v_y/|{}^{I}v_{cmd}|) \tag{4.13}$$

where the components of ${}^{I}v_{cmd}$ are $[v_x, v_y, v_z]^T$. Note that the angles θ and ψ define the commanded velocity vector's orientation in $\{I\}$.

Finally, the amount that V_{t-cmd} is outside of indicated airspeed limits is subtracted from the magnitude of ${}^{I}v_{cmd}$, resulting in the magnitude of the new commanded inertial velocity, termed V. V is then resolved in $\{I\}$ according to:

$${}^{I}v_{mod} = \begin{bmatrix} cos(\theta)cos(\psi) & -cos(\theta)sin(\psi) & -sin(\theta) \\ sin(\psi) & cos(\psi) & 0 \\ sin(\theta)cos(\psi) & -sin(\theta)sin(\psi) & cos(\theta) \end{bmatrix} V \tag{4.14}$$

where ${}^{I}v_{mod}$ is the new commanded inertial velocity. This command is integrated and sent to the controller as the commanded trajectory. The *MATLAB .m file* that implements this logic is *windlogic.m* and can be found in Appendix A.

The net effect of the trajectory logic block is simple. When Bluebird runs up against one of its airspeed limits, the commanded trajectory can no longer be followed. A choice is made to maintain the direction of the commanded velocity but change its magnitude. Notice that this method does not affect the turn rate associated with the trajectory and subsequently, no processing of the angular trajectory commands is required. In this way, Bluebird is never commanded to fly a trajectory that would force it to exceed the performance limits.

The performance of the trajectory logic block can be seen in Figure 4.4. The lower limit for Bluebird was arbitrarily choosen as 63 feet per second. Bluebird is initially flying due north at a ground speed of 73 feet per second, crabbed into 20 feet per second of wind from the west. The commanded trajectory turns 90 degrees to the east. Notice that the original trajectory would result in an indicated airspeed of 53 fps while the revised trajectory results in a commanded trajectory of 63 fps.

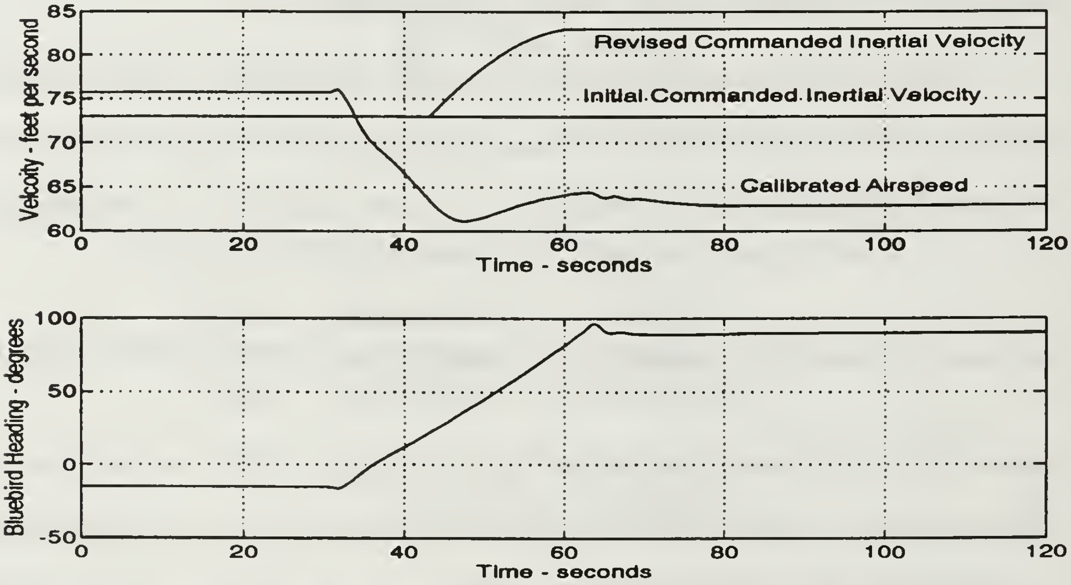


Figure 4.4: Example of Commanded Trajectory Revision

If the commanded trajectory is generated using a velocity rather than position schedule, the differentiation block in Figure 4.4 can be removed. A velocity schedule is specified in $\{I\}$ as a sine wave of appropriate magnitude, frequency, and phase along the x and y axis. The commanded ground speed corresponds to the magnitude and the commanded turn rate corresponds to the frequency of the sine function. Constant heading flight is a subset of these trajectories where the frequency is zero. This method of generating trajectory commands makes determining turning rate ($\dot{\Lambda}_{cmd}$) simple. As an example, consider the case of generating the velocity schedule for a circular flight pattern at a ground speed of 100 fps and a turn rate of 0.1 radians per second. The derivative of the commanded trajectory, $\dot{r}$, parameterized by time is:

$$\dot{r} := \begin{cases} x\ axis\ velocity & = 100 sin(0.1t + pi/2) \\ y\ axis\ velocity & = 100 sin(0.1t + 0) \\ z\ axis\ velocity & = desired\ climb\ or\ descent\ rate \\ \dot{\Phi} & = 0 \\ \dot{\Theta} & = 0 \\ \dot{\Psi} & = 0.1 \end{cases} \tag{4.15}$$

The SIMULINK block that generates the commanded trajectory is shown in Figure 4.5.

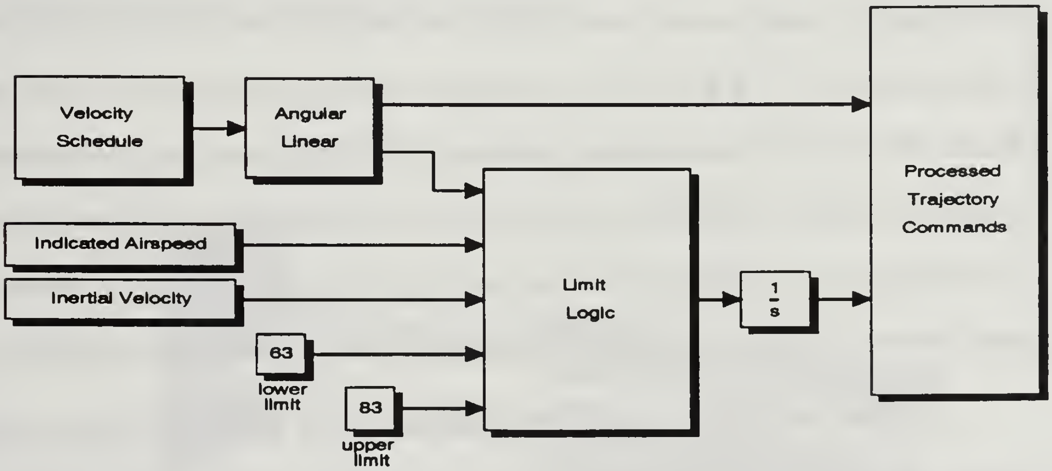

Velocity Schedule
Angular Linear
Indicated Airspeed
Inertial Velocity
63
lower limit
83
upper limit
Limit Logic
1/s
Processed Trajectory Commands

Figure 4.5: Generation of Trajectory Commands

D. STATE FEEDBACK TO OUTPUT FEEDBACK

Up until this point, the development of the tracking controller has dealt exclusively with full state feedback. This section will detail a method whereby the full state feedback is replaced by high fidelity models of the onboard sensors in conjunction with Kalman filters designed to provide optimal state estimates, using onboard sensor data.

1. Sensor Modeling

This work builds upon sensor models developed in two prior theses. In [Ref. 1], a detailed model of the inertial measurement unit, IMU, is developed. In [Ref. 2], a detailed model of the GPS unit is developed. The IMU is a compact, lightweight, low power unit which integrates nine sensor measurements in one box. These sensors are three axis accelerometers, three axis rate gyros, pitch and roll inclinometers, and a magnetometer. The accelerometers are instrument grade, signal conditioned, and temperature compenstated. Full scale output is $^+/-3$ g's. The accelerometer's frequency response is flat past 100 Hz. However, the antialiasing inside the IMU limits the effective bandwidth of all of the sensors to 20 Hz. An internal initialization program allows the unit to compenstate for accelerometer bias and cross axis error. Table 4.1 shows the specifications of the accelerometers incorporated into the sensor model [Ref. 8].

The rate gyros used by the IMU are solid state vibrating element angular rate sensors. This relatively new technology uses no moving parts. A piezoelectric bender element is mounted end to end but rotated at a 90 degree angle. The element fastened to the base is resonantly driven such that the sense element swings a reciprocating arc. Under zero angular rate conditions, the motion of the sense element due to the drive element does not produce any

TABLE 4.1: ACCELEROMETER CHARACTERISTICS

Acceleration Range	$\pm 2g's$
Acceleration Bandwidth	20 Hz
Acceleration Bias	0.2% of Full Scale
Acceleration Scale Factor	0.2% of Full Scale
Acceleration Noise Floor	0.0005 $g's$
Cross Axis Sensitivity	0.5% of Full Scale

bending of the sense element. When a rate of rotation exists, Coriolis forces cause momentum to be transfered into the plane perpendicular to the motion of the drive element, thus causing bending of the sense element. A pressure transducer picks up a signal from the sense element when it is bent that is proportional to the angular rate with a phase dependent on the direction of rotation. Figure 4.6 shows the configuration of two rate sensors mounted in a "tuning fork" configuration. Table 4.2 shows specifications of the rate sensors incorporated into the sensor model [Ref. 8].

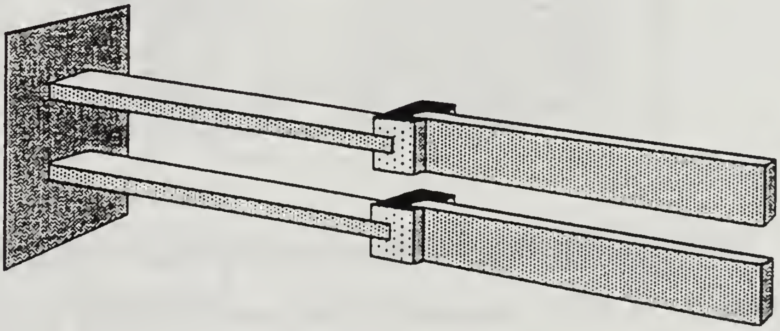


Figure 4.6: Angular Rate Sensor

TABLE 4.2: ANGULAR RATE SENSOR CHARACTERISTICS

Rotational Rate Range	$\pm 114.6 deg/sec$
Rotational Rate Bandwidth	20 Hz
Rotational Rate Bias	2.0% of Full Scale
Rotational Rate Scale Factor	0.5% of Full Scale
Rotational Rate Noise Floor	0.05% of Full Scale
Cross Axis Sensitivity	0.5% of Full Scale

The inclinometers utilize a liquid crystal pendulous sensor. It is a low bandwidth sensor (approximately 0.12 radians per second) that is meant to be integrated with the rate sensors for high bandwidth measurements of angular position. The fluxgate magnetometer provides heading measurements. The specifications incorporated into the Euler angle sensor models are shown in Table 4.3 [Ref. 8].

TABLE 4.3: INCLINOMETER AND MAGNETOMETER CHARACTERISTICS

Pitch and Roll Range	± 50 *deg*
Pitch and Roll Bandwidth	1/2 Hz
Pitch and Roll Accuracy	0.2 *deg*
Heading Range	± 180 *deg*
Heading Accuracy	3.0 *deg*
Heading Repeatability	0.5 *deg*
Heading Linearity	0.5%

GPS provides data in a form that can be converted to local tangent plane position. A brief summary of the errors included in the GPS model follows.

GPS Error Sources

- Selective Availability

  - intentional degradation of pseudorange signal by Departmant of Defense

- Clock Differences
  - drift and bias in GPS clock
- Ephemeris Error
  - error introduced in converting pseudoranges to inertial position fix

Each of these sensor components is simulated in block diagram form in SIMULINK utilizing internal modeling principles based on manufacturer specifications and known sources of error. The upper level SIMULINK diagram of these sensor models is shown in Figure 4.7 and contained in the SIMULINK file *plant1.m.*

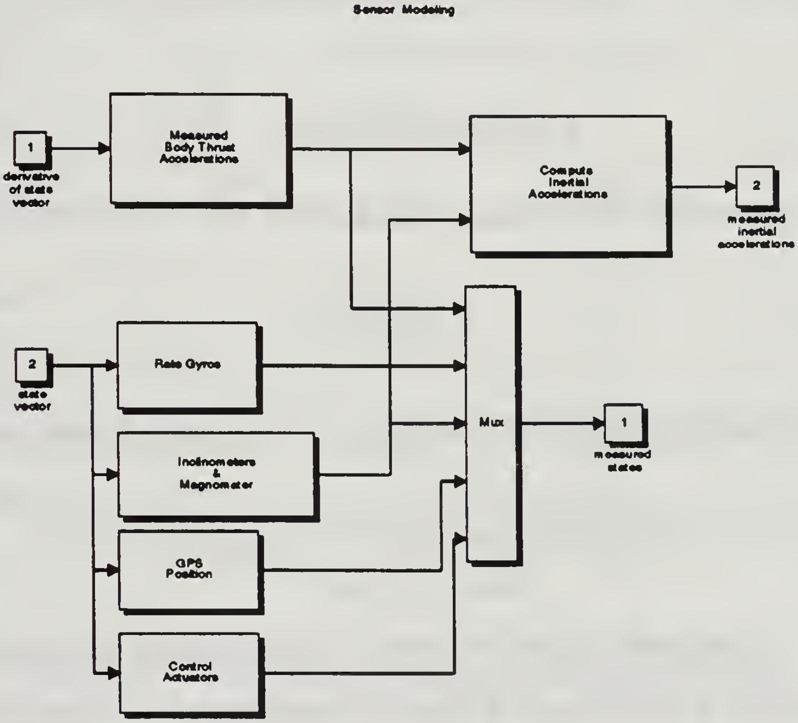

Sensor Modeling
1
derivative of state vector
Measured Body Thrust Accelerations
Compute Inertial Accelerations
2
measured inertial accelerations
2
state vector
Rate Gyros
Inclinometers & Magnometer
GPS Position
Control Actuators
Mux
1
measured states

Figure 4.7: Sensor Models in SIMULINK

2. Kalman Filtering

With the outputs of the modeled sensors available, Kalman filters were developed along the linear and angular position channels to provide optimal estimates of the states for the controller. An approach analogous to the LQR design was used. Consider the linear system described by:

$$\begin{aligned} \dot{x} &= Ax + Bu + Gw \\ z &= Cx + v \end{aligned} \tag{4.16}$$

where v and w are zero mean white noise with respective power spectral densities of V and W.

A gain matrix, L, was found such that the Kalman filter given by:

$$\dot{x} = Ax + Bu + L(z - Cx) \tag{4.17}$$

produces an optimal estimate of x. The Kalman gain L is calculated as follows:

$$L = YC^T V^{-1},$$

where Y is positive semidefinite and solves the algebraic Riccati equation:

$$AY + YA^T - YC^T V^{-1} CY + GWG^T = 0$$

A synthesis model was formed that included the dynamics of the original plant. The IMU used in Bluebird incorporates an initialization routine that removes steady state bias from the sensors. Therefore, extra dynamics were not required in the Kalman filter to compensate for bias. The design process was primarily driven by the bandwidth limitations of the inclinometers and GPS. The values of V and W were used as "knobs" in the iterative design process.

For the GPS sensor, a break frequency of 1/2 radian per second was desired. For the Euler angle sensors, a break frequency of 1/10 radian per second was desired. It was also desired to wash out the accelerometers and rate sensor biases at low frequencies.

The Kalman filter for the position estimate blends processed accelerometer outputs with the GPS position fixes. Figure 4.8 shows the frequency response along the x axis of the Kalman position filter. The other two axes have similar dynamics.

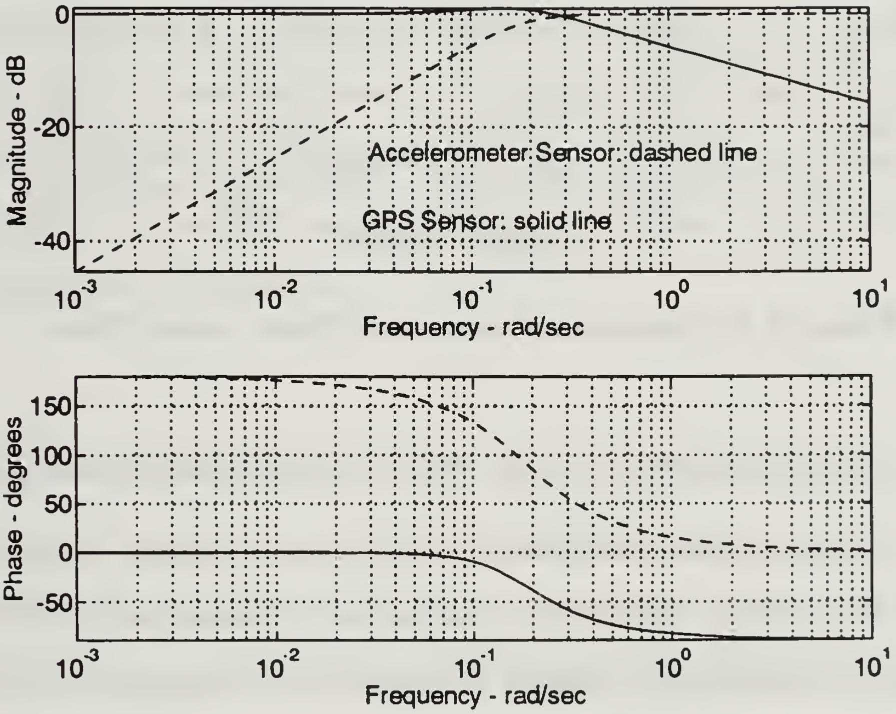


Figure 4.8: Frequency Response of Position Filter

The Kalman filter for the Euler angles blends the outputs of the angular rate sensors, converted to Euler angle rates, with Euler angle measurements from the inclinometers and magnetometer. Figure 4.9 shows the frequency response of one channel of the Euler angle filter. The remaining two angle channels are similar.

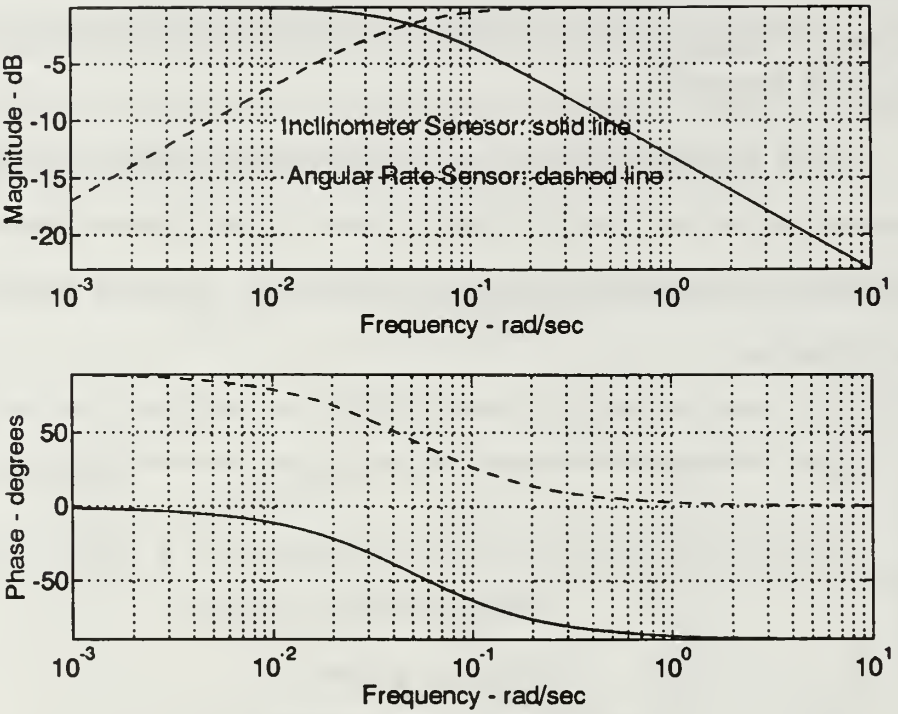


Figure 4.9: Frequency Response of Euler Angle Filter

E. INTEGRATION OF THE FULL NONLINEAR SIMULATION

The full nonlinear simulation can now be pieced together. Recall in Chapter II, the nonlinear rigid body dynamics were implemented in a SIMULINK block labeled *Equations of Motion.* If the simulation is expanded to include the effects of a moving airmass, the dynamics of Bluebird can be simulated at an arbitrary flight condition. Wind is usually referenced with respect to the inertial reference frame, therefore a SIMULINK block, *Wind*, is included in the full simulation whose output is a vector w_v comprised of the wind velocity resolved in $\{I\}$. Next, the wind velocity is resolved in $\{B\}$ and added to the inertial velocity of Bluebird, ${}^{B}v_{cg}$. The result is the velocity of Bluebird with respect to the airmass, resolved in $\{B\}$. This velocity, rather than ${}^{B}v_{cg}$, is used when computing the aerodynamic contribution to the total external force and moment used in Equation 2.25. A more detailed description of how this is accomplished in the .m file *state_deriv.m* is contained in Appendix A.

All sixteen states are sent to a SIMULINK block, *Sensors,* that models the avionic sensor suite onboard Bluebird. The output from these sensors is appropriately processed in a SIMULINK block, *Kalman filters.* The filtered output is directed to the *$\mathcal{D}$-Implemented Controller* block. The commands to the controller come from a trajectory block which uses the measured outputs from the filter block to process the commanded trajectory. The controller generates actuator commands necessary to maintain the air vehicle on the commanded trajectory, thus completing the loop. The complete top level view of the SIMULINK nonlinear simulation is shown in Figure 4.10 and contained in the SIMULINK file *plant1.m.*

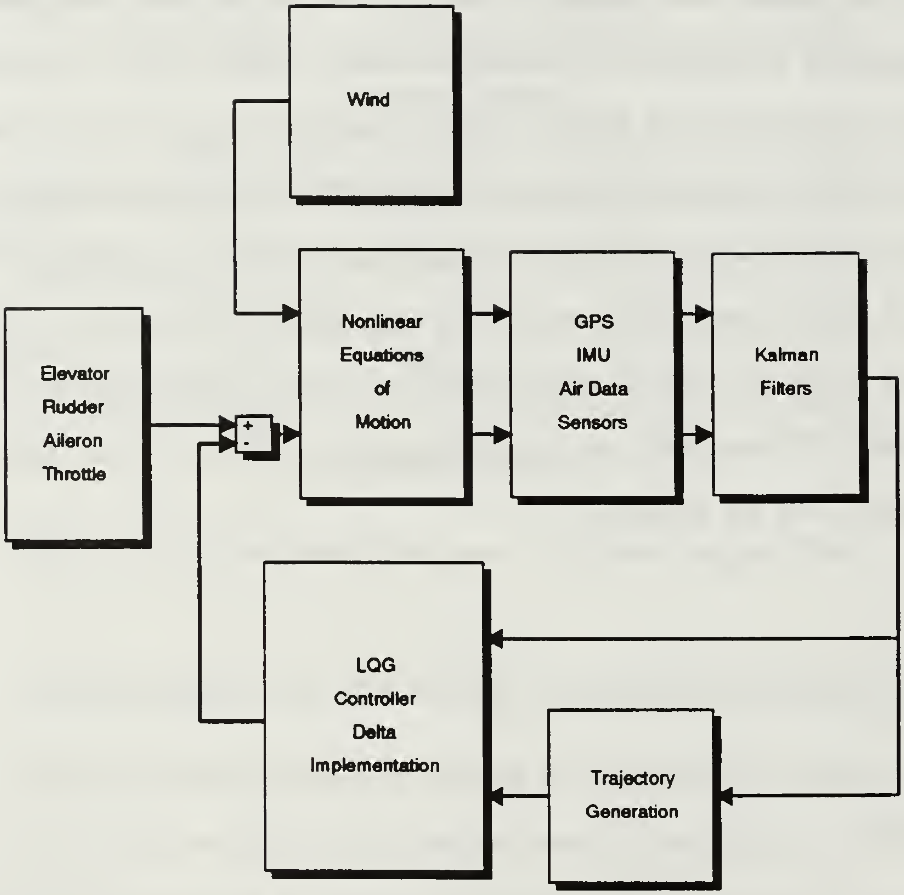

Wind
Nonlinear Equations of Motion
GPS IMU Air Data Sensors
Kalman Filters
Elevator Rudder Aileron Throttle
+
-
LQG Controller Delta Implementation
Trajectory Generation

Figure 4.10: SIMULINK Diagram of the Full Nonlinear Simulation

V. APPLICATION TO THE CONTROL OF BLUEBIRD

A. $\mathcal{D}$-IMPLEMENTED CONTROLLER PERFORMANCE CHARACTERISTICS

The performance characteristics of the trajectory tracking controller designed in Chapters III and IV were evaluated using a two step process. First, the *sensor* and *filter* blocks were removed and the controller was connected directly to the nonlinear *equations of motion* block. Utilizing pure state feedback, Bluebird was flown along two fundamentally different types of trajectories. These two trajectories served as general examples of the set of all trajectories defined in Chapter IV, Equation 4.4. Next, the *sensor* and *filter* blocks were added. A general deparure and arrival trajectory, which is a combination of the two types of trajectories tested in the first step, was commanded and flown with the controller utilizing output feedback. Data from this simulation were used as input to a virtual prototype simulation discussed later.

The dynamic flight simulations were started using the same initial condition. At this initial condition, Bluebird is aligned with the positive x-axis and trimmed for level flight at 73 fps. The positive x direction is considered to be heading north. The mechanics of the dynamic simulation use a right-hand orthogonal coordinate system described in Chapter II. As such, the positive y-axis is pointing east and the positive z-axis is pointing down. This choice is convenient from a computational standpoint since it coincides with the body-fixed reference frame of Bluebird at the initial condition specified above. Typically, however, the positive z axis is considered to be pointing up. If the right-hand orthogonality is maintained, the positive y-axis would then point towards the west. For ease of visualization, this is the cooridinate system used to display the results of the simulations.

The simulations are intended to evaluate the capabilities of the controller in terms of the nature of trajectories that can be followed in a stationary air mass and in an air mass moving at constant velocity. Two basic kinds of trimmed flight serve as the bases for the test trajectories. Each test trajectory is flown in no-wind conditions, and then again with the wind added at some point during the flight.

The simplest form of trimmed flight is constant velocity, constant heading. This corresponds to a trajectory defined by a ramp command in inertial position. This was the basic trajectory that the controller was designed to track. Figure 5.1 shows a three dimensional plot of the first test trajectory. In this case the trajectory encompasses 30 seconds of flight heading north at 73 fps followed by a 90 degree turn to join a trajectory heading east at 73 fps while climbing at 300 feet per minute. On the second flight, a wind of 20 feet per second from the north is added at the time the turn is commanded (elapsed time = 30 seconds).

Figure 5.2 contains the first four graphs of flight data. The first graph shows the time history of Bluebird's distance from the commanded trajectory. The next three graphs show the time history of the Euler angles. Consider the baseline flight (no-wind data). Bluebird begins the turn at an elapsed time of 30 seconds and exits the turn at an elapsed time of 45 seconds. Approximately 30 seconds later the trajectory error is nearly zero. In the presence of 20 feet

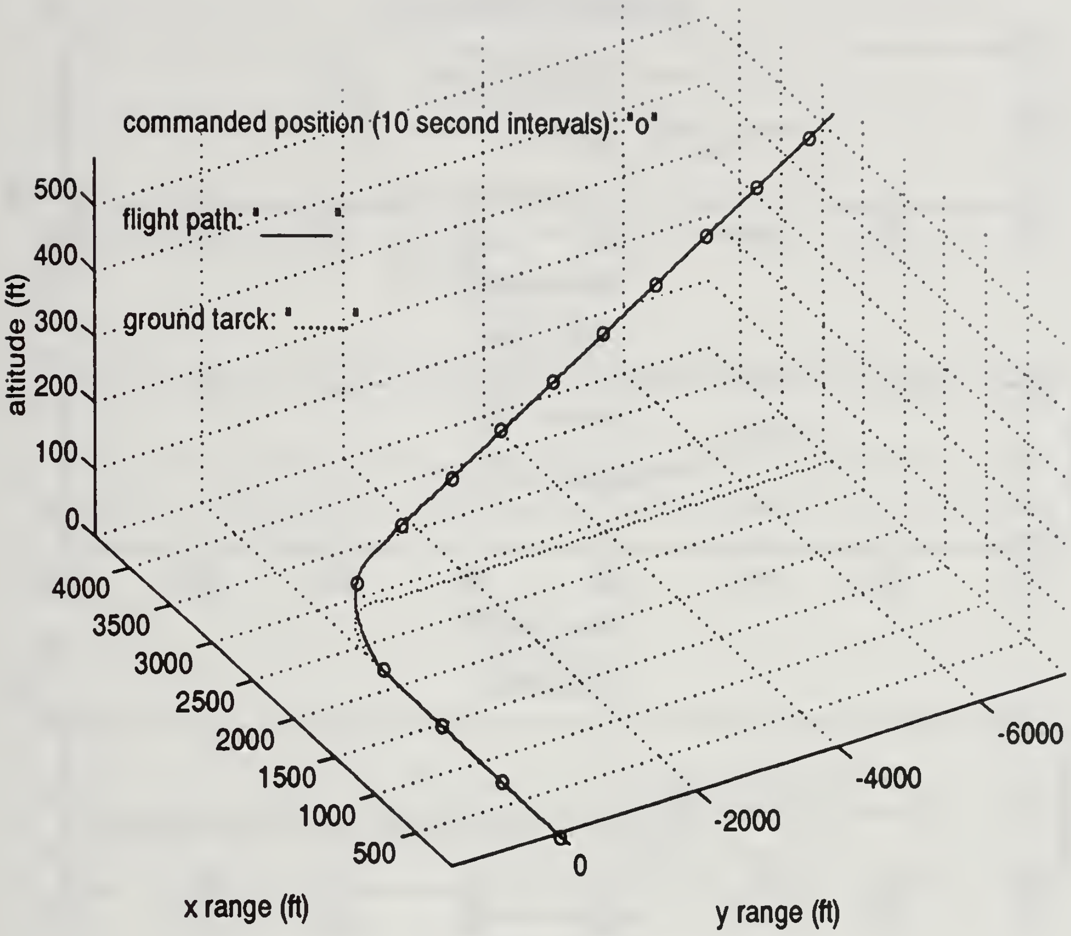


Figure 5.1: Test Trajectory #1

per second of wind from the north, the trajectory error also goes to zero in about the same period of time. The graphs of the Euler angles indicate that Bluebird is flying wings level, crabbed into the crosswind, which is the desired result. Figure 5.3 shows the groundspeed and indicated airspeed during the flights. Notice that in both cases, the commanded groundspeed of 73 feet per second is eventually maintained. Finally, Figure 5.4 shows the time history of the control activity during the flights.

Trimmed flight does not necessarily have ${}^{B}\omega_{B}$ equal to zero. For instance, any steady turning manuever fits the definition of trimmed flight given in Chap-

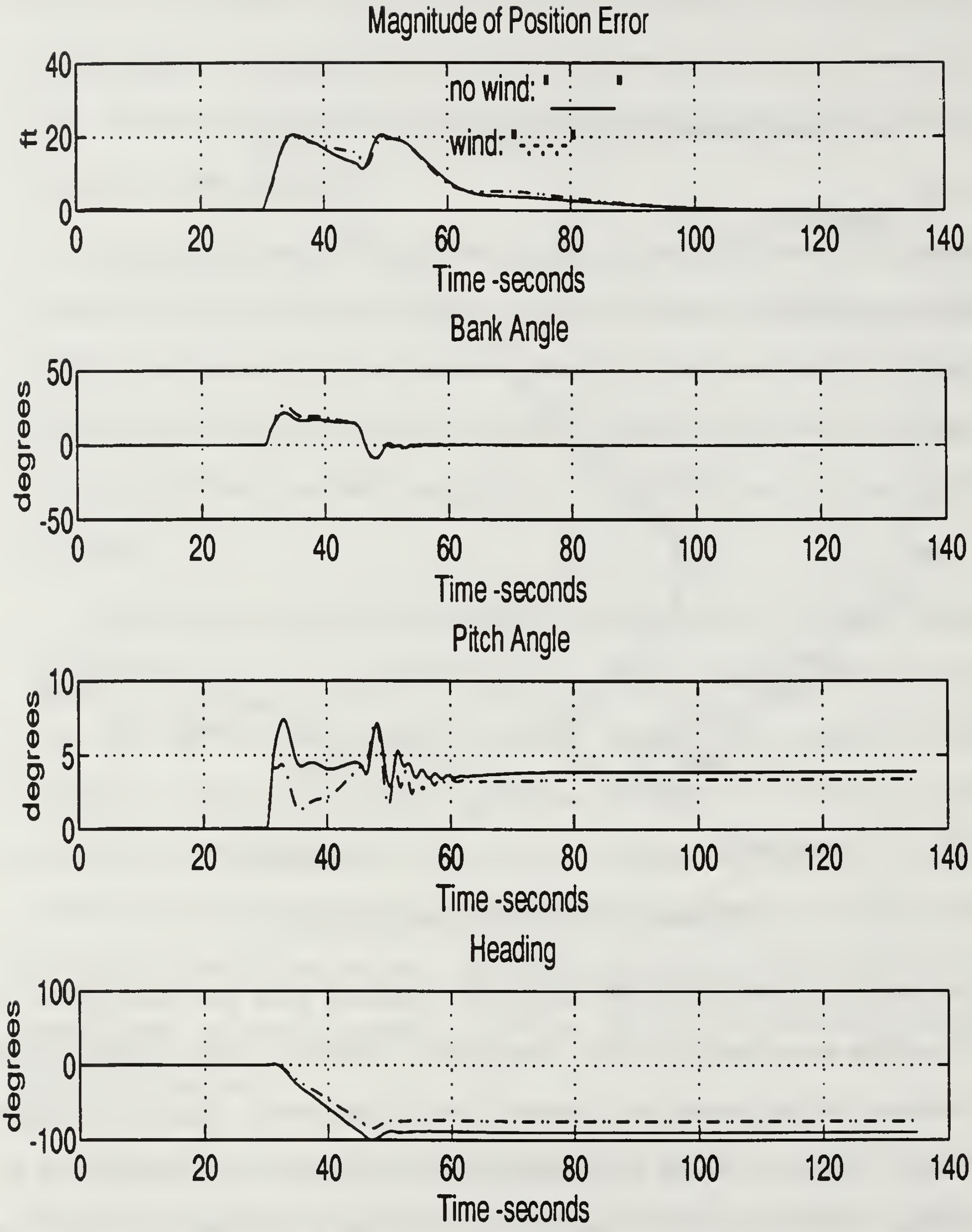


Figure 5.2: **Trajectory #1: Position Error and Euler Angles**

ter IV. Figure 5.5 shows the three dimensional plot of the second test trajectory. In this case the test trajectory is a helix flown at 73 feet per second. The turn rate is one revolution per minute and the helix angle is 4 degrees which corre-

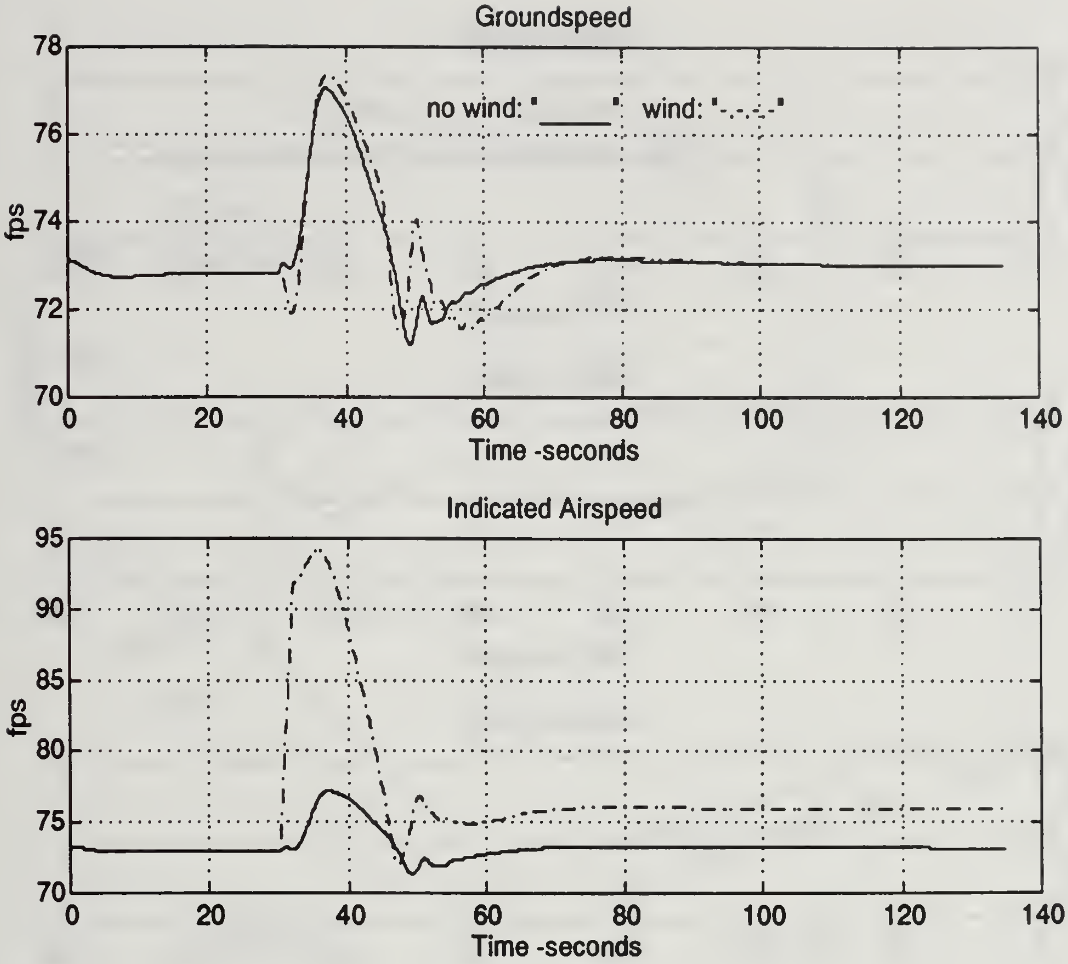


Figure 5.3: Trajectory #1: Velocity Data

sponds to a climb rate for Bluebird of 300 feet per minute. For this test, no indicated airspeed limits were placed on Bluebird.

Consider Figure 5.6 which shows the position error and Euler angle time history for the helix trajectory. Notice that with no wind the controller maneuvers Bluebird to join the commanded trajectory with zero error in steady state. The constant pitch and bank angles confirm the steady state performance. On the second flight, a wind of 20 feet per second from the east was added at the start of the helical trajectory (elapsed time equal to 40 seconds). Intuitively, it is clear that the bank and pitch angle must vary as Bluebird traverses the

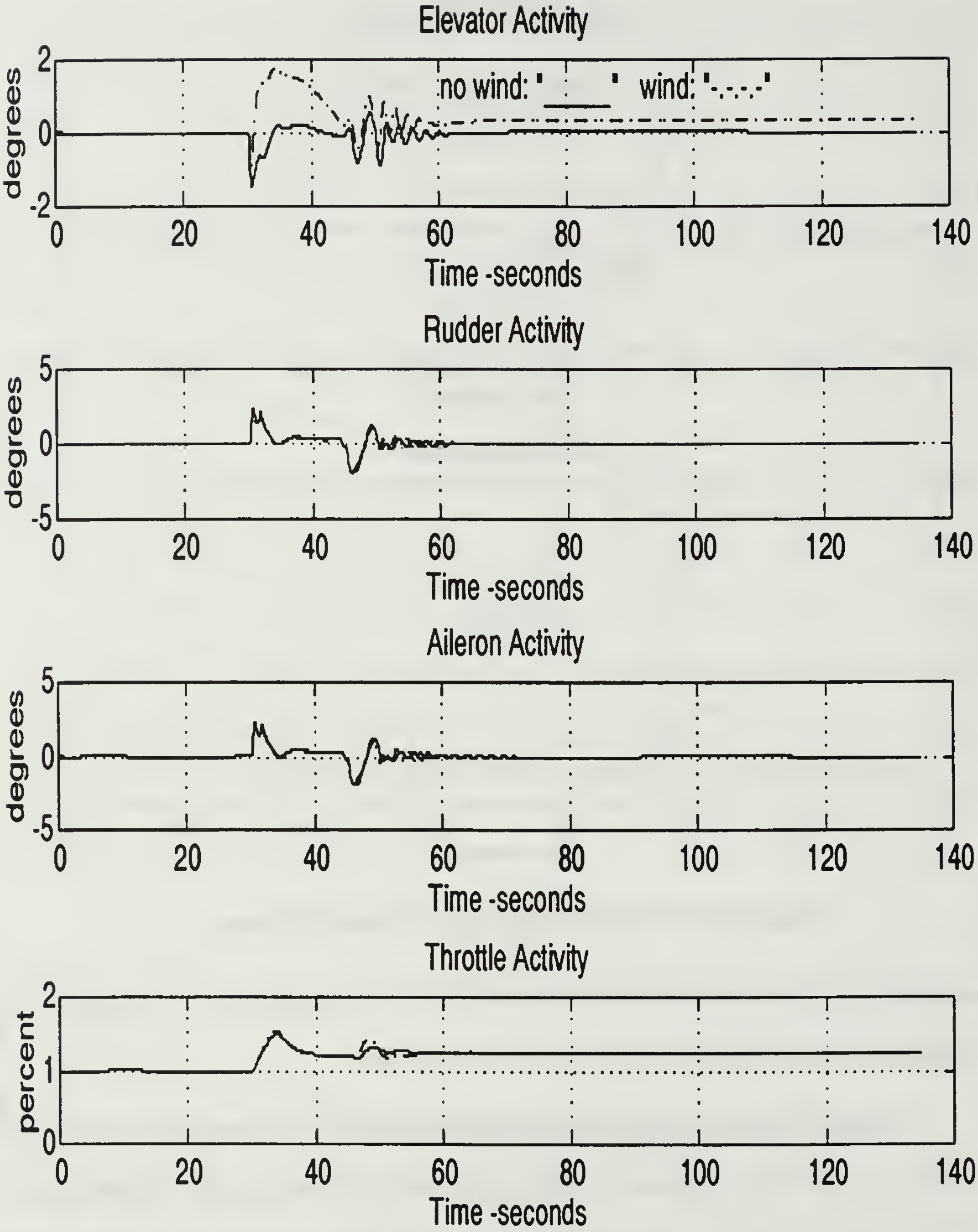


Figure 5.4: Trajectory #1: Control Activity

helix. To an observer on Bluebird, the wind, while constant in $\{I\}$, represents a sinusoidal disturbance. This explains the sinusoidal nature of the position error around the helix. Figure 5.7 shows the groundspeed and indicated airspeed

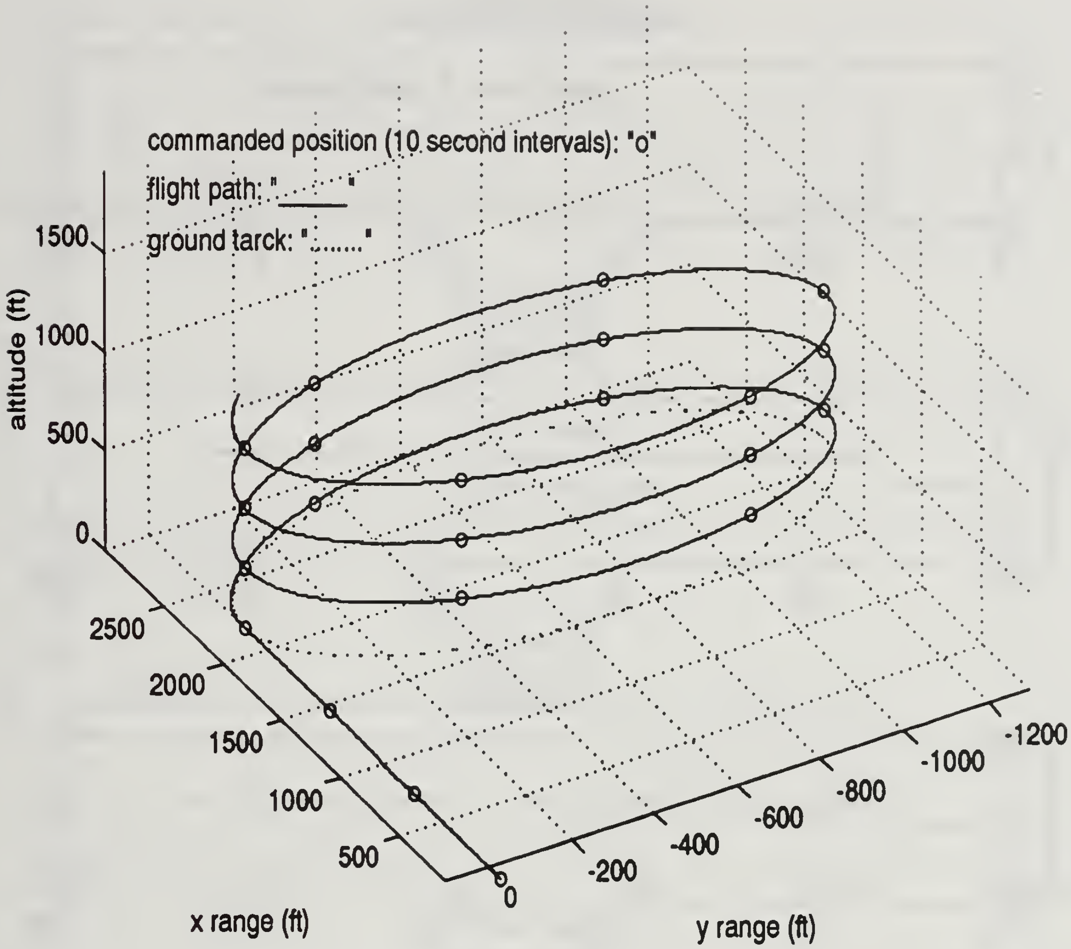


Figure 5.5: Test Trajectory #2

around the helix during the two flights. For the no-wind flight, the commanded groundspeed is maintained while for the flight in wind, a one foot per second oscillation is experienced. Finally, Figure 5.8 shows the time history of the control activity during the flights.

Four additional flights were flown using the helix trajectory in an attempt to ascertain the sensitivity of the position error to changes in commanded turn rate and wind velocity. Figure 5.9 shows the position error around the helix for three different turn rates with a constant wind of 10 feet per second. The dashed line corresponds to a turn rate of 3 degrees per second or 2 minutes

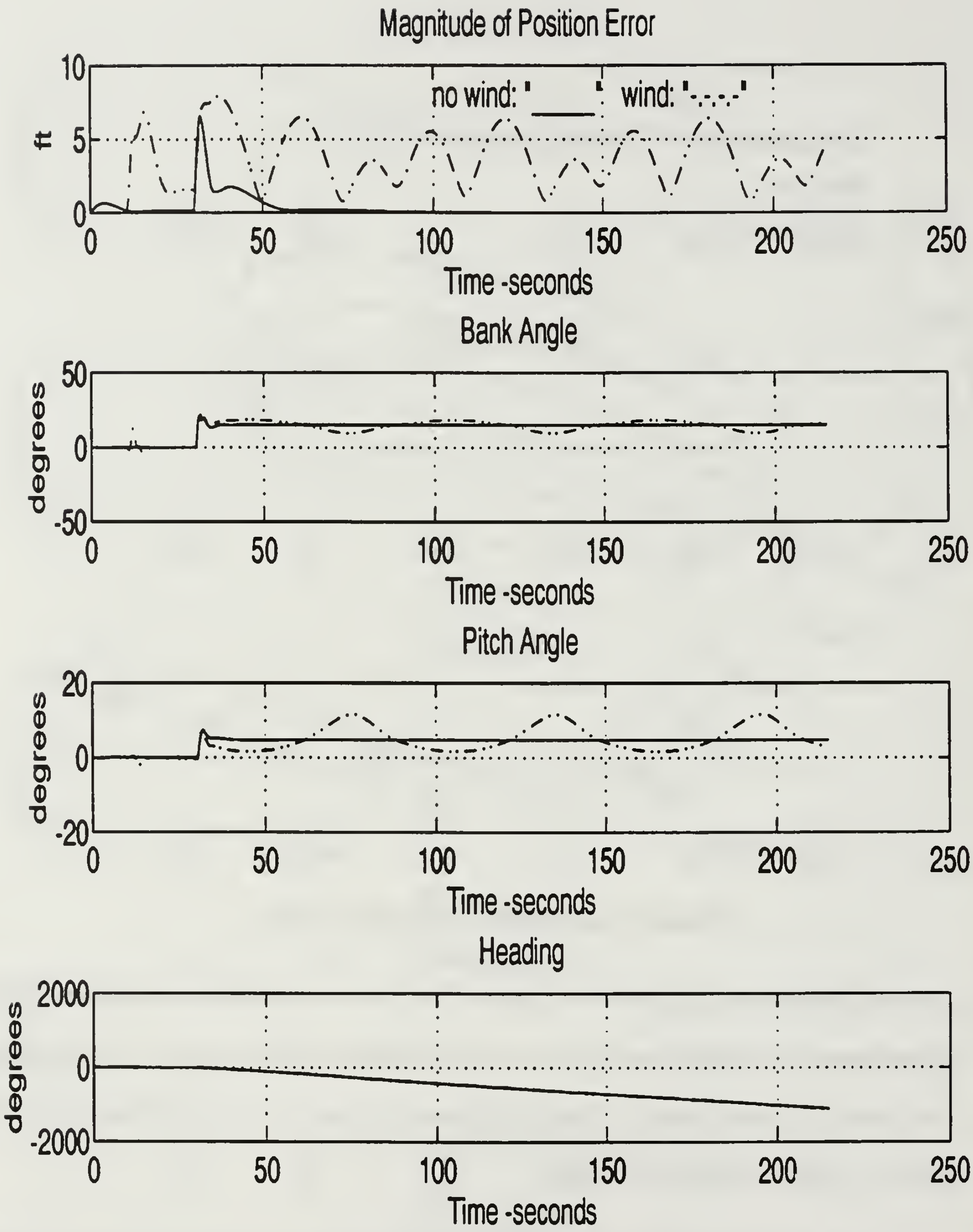


Figure 5.6: Trajectory #2: Position Error and Euler Angles

per revolution; the solid line corresponds to a turn rate of 6 degrees per second or 1 minute per revolution; and the dash dot line corresponds to a turn rate of 9 degrees per second or 40 seconds per revolution. The error increases with

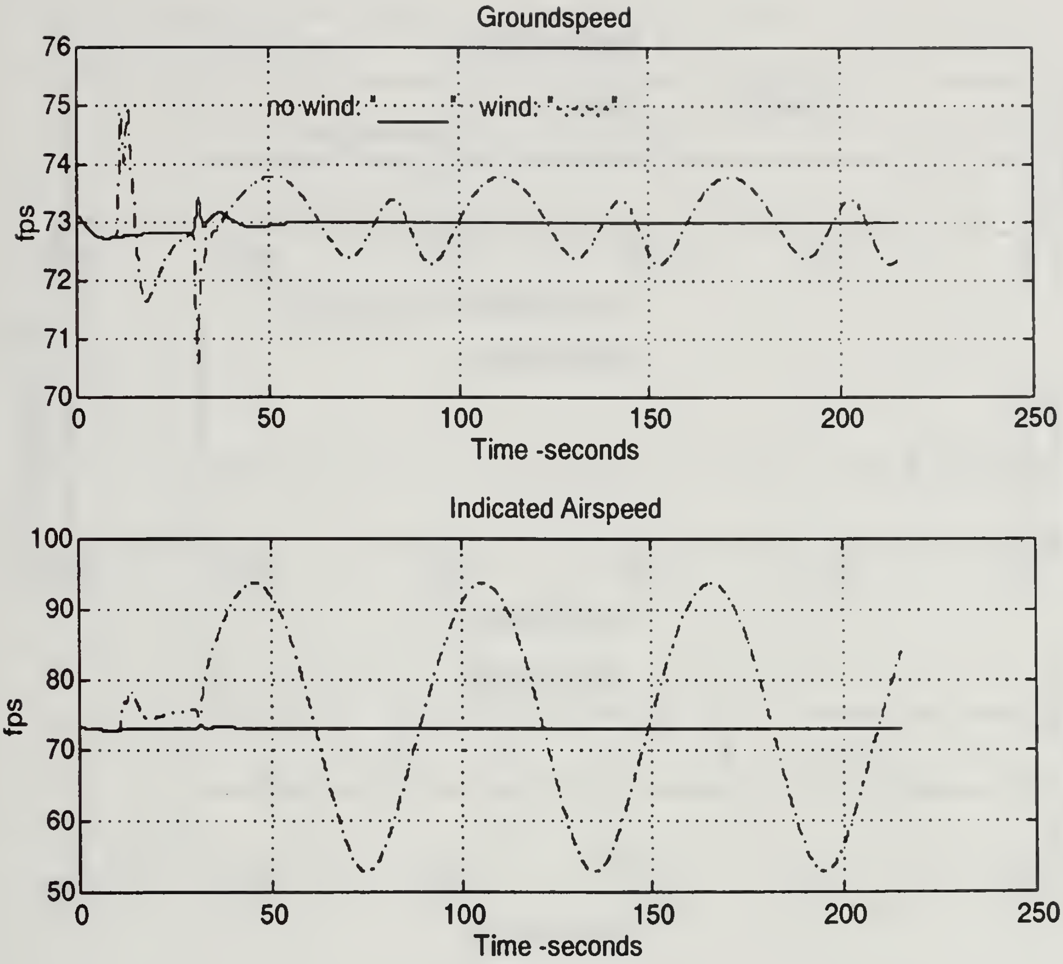


Figure 5.7: Trajectory #2: Velocity Data

increasing turn rate. The 9 degree per second turn rate corresponds to a steady state angle of bank of thirty degrees, no wind. It may not be desireable to command trajectories requiring more than a certain angle of bank and this may place an upper bound on the error.

Figure 5.10 shows the position error around the helix for three different wind velocities at a constant turn rate of 6 degrees per second. The wind varies in velocity from 10 to 25 feet per second. The local maxima values of the position error are proportional to the magnitude of the disturbance.

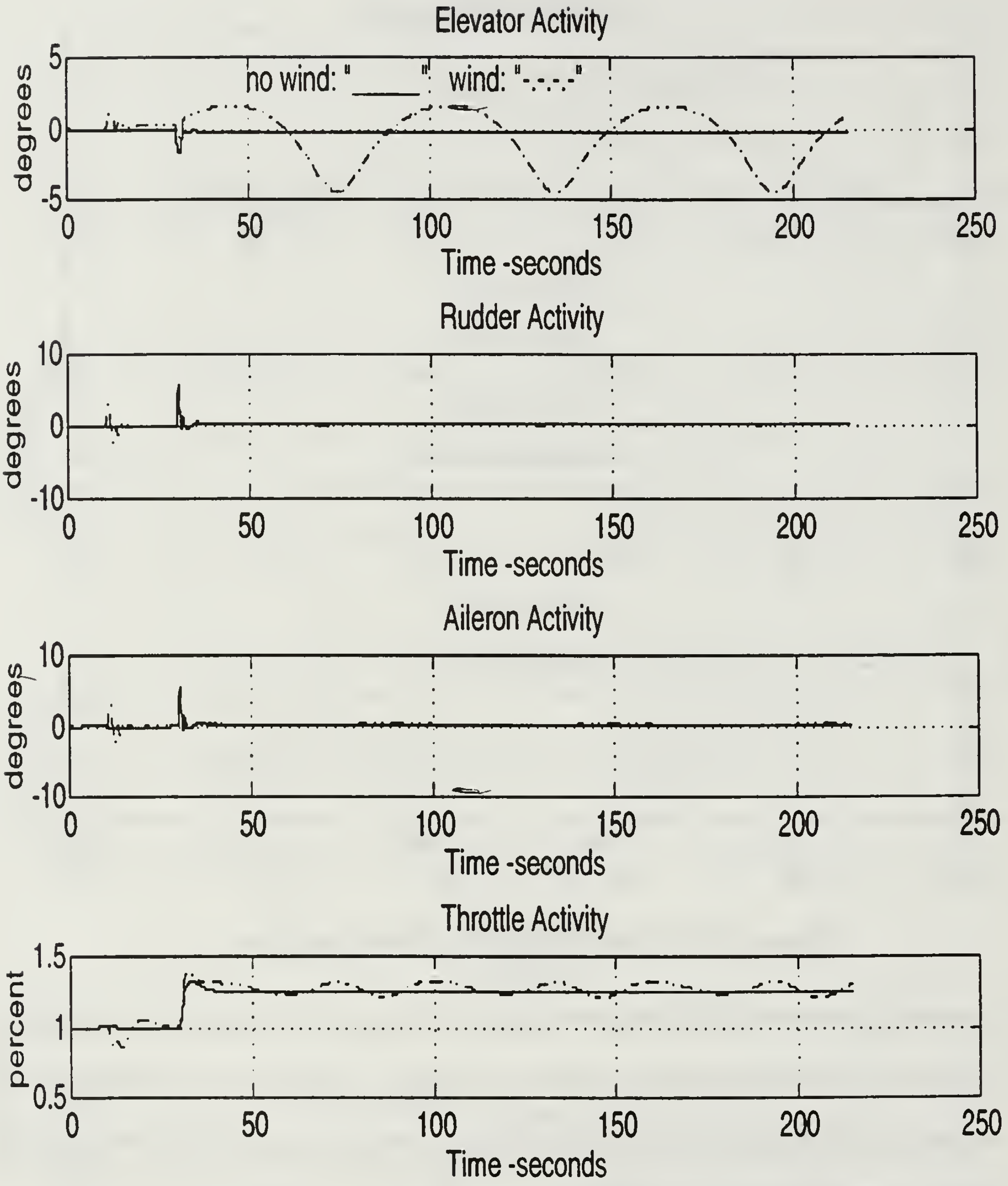


Figure 5.8: Trajectory #2: Control Activity

B. AN AIRPORT DEPARTURE AND ARRIVAL FLIGHT SIMULATION

In many cases, the trimmed flight condition of an air vehicle changes often. A good example of this would be a standard instrument departure or arrival

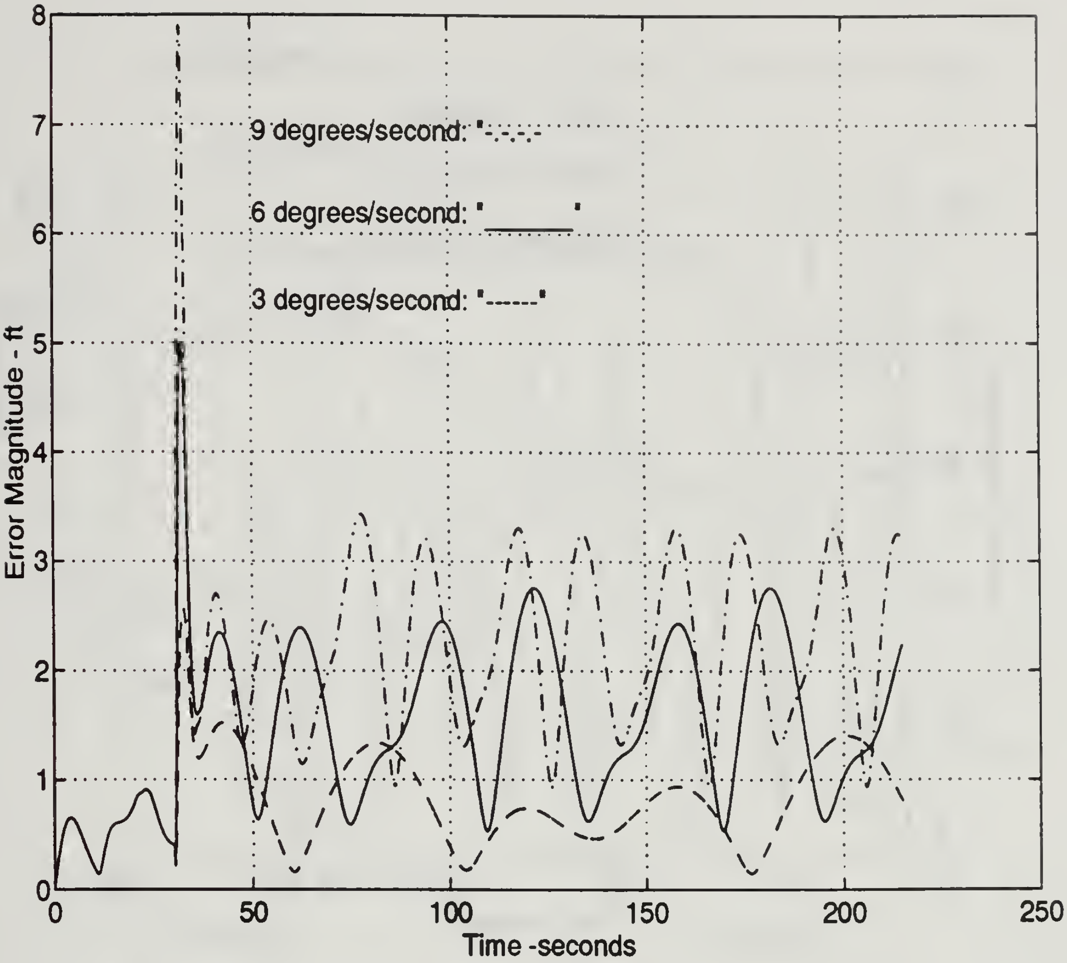


Figure 5.9: Trajectory #2: Position Error for Varying Turn Rates

to an airfield. These trajectories are typically a combination of constant radius turns and wings level flight, while often climbing and descending. Trajectory position errors become critical when the air vehicle is on final approach with a constant heading, constant velocity trajectory.

Consider Figure 5.11. If an airfield is imagined to be located at the origin, then this trajectory would be indicative of a typical departure followed by a typical arrival to that airfield. The scenario utilizes turning trajectories of three different radii connected by straight line trajectories. The commanded velocity is a constant 73 feet per second throughout. Wind is initially zero. Thirty

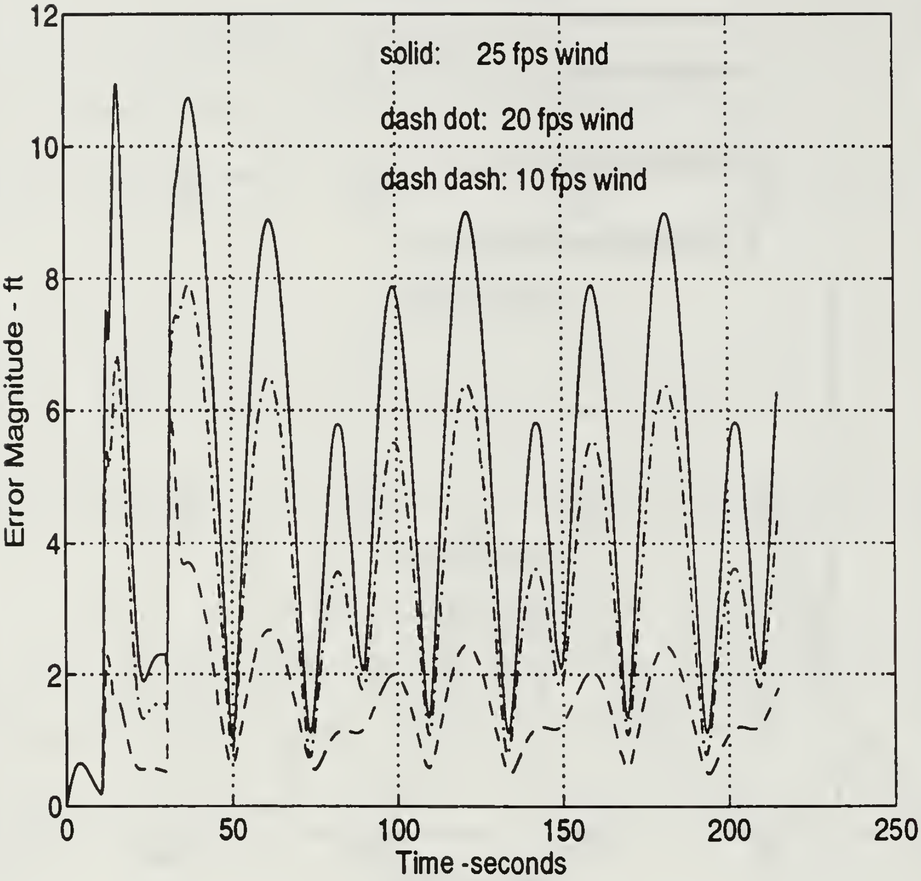


Figure 5.10: Trajectory #2: Position Error for Varying Wind Velocities

seconds into the flight, the wind is added at 10 feet per second from the east. At 90 seconds into the flight the wind is increased to 20 feet per second from the east. Finally, with Bluebird on final approach tracking a 4 degree glideslope, the wind is rapidly shifted 90 degrees to the north and decreased in magnitude to 5 feet per second.

Figure 5.12 shows the time history of the position error, wind velocity, and Euler angles during the flight. Figure 5.13 shows the time history of the control activity. Note, however, the relative difficulty of analyzing data of this nature as it is somewhat difficult to visualize. Flight simulation data was saved to a file

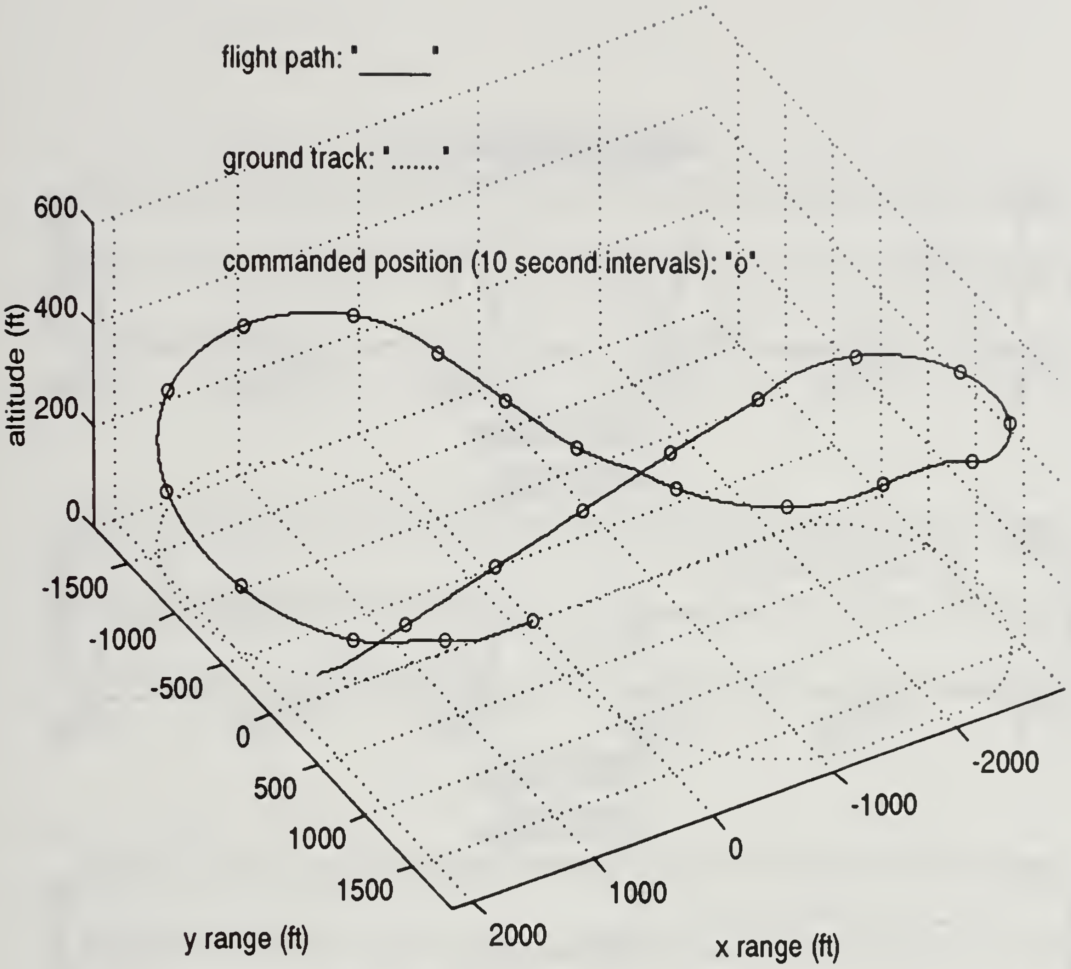
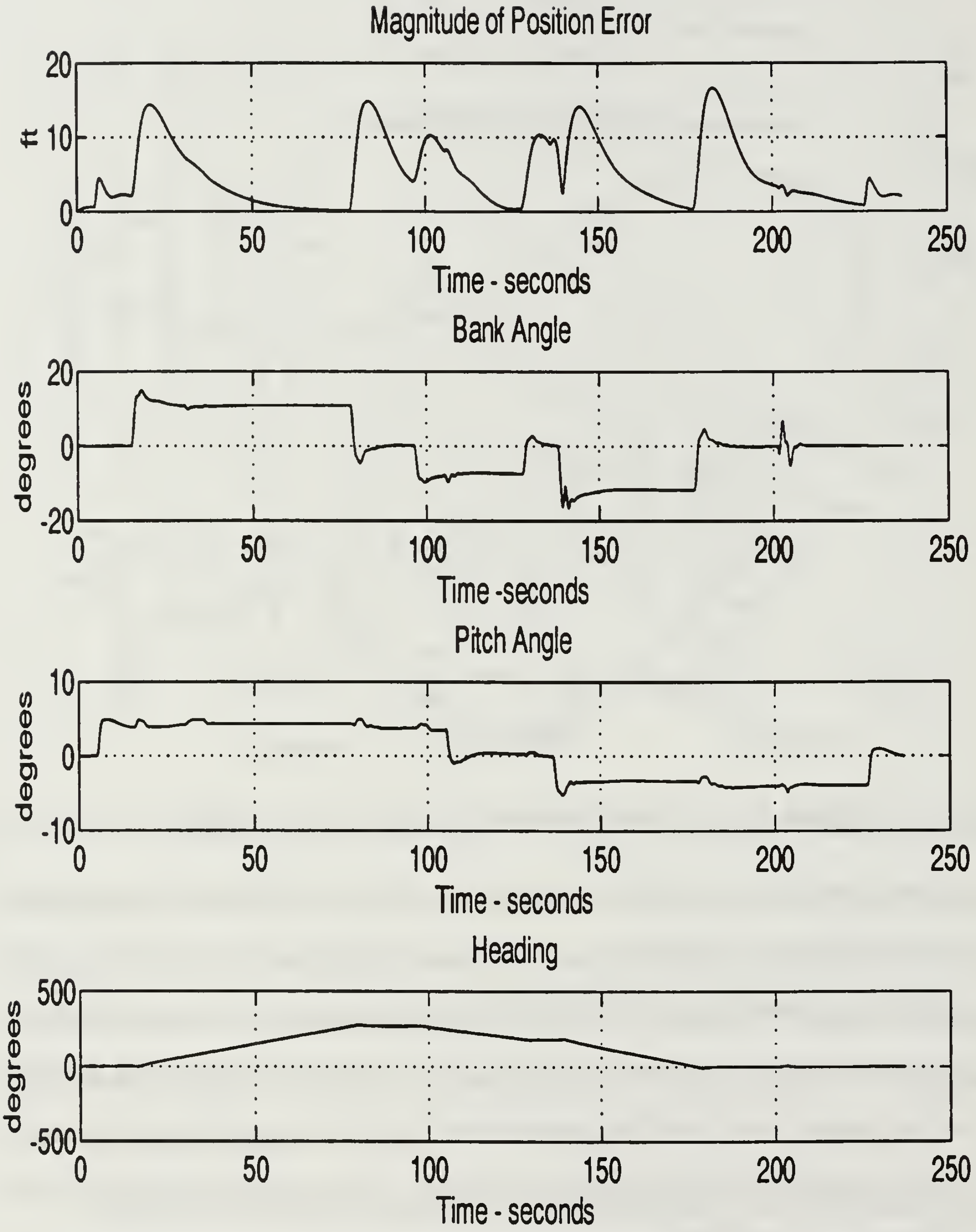


Figure 5.11: Departure and Arrival at an Airfield

and processed for compatiblity as an input file for a 3-D visualiztion software package, Designer's Workbench. A virtual prototype of Monterey Airport and Bluebird was developed in [Ref. 13]. The simulation was then run as a departure and arrival to the virtual prototype airfield. In Designer's Workbench, the flight can be viewed from multiple perspectives and virtual prototypes of standard cockpit displays further enhance visualization. One possible result is captured on video tape, [Ref. 14].


Magnitude of Position Error

Bank Angle

Pitch Angle

Heading

Figure 5.12: Airfield Scenario: Position Error and Euler Angles

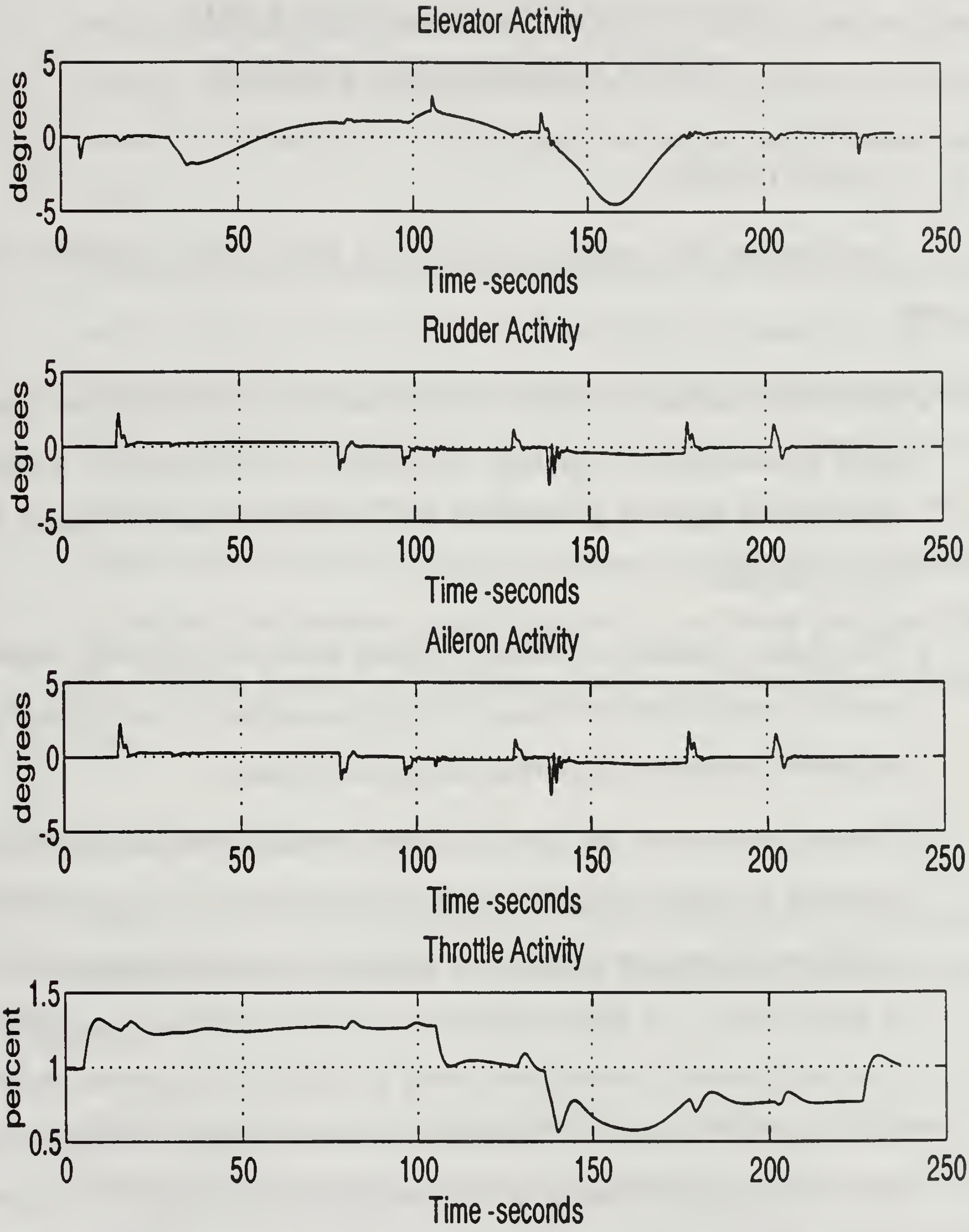


Figure 5.13: Airfield Scenario: Control Activity

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Based on the data presented in this thesis, the following conclusions are drawn.

- SIMULINK provides an effective environment for the developement of nonlinear simulations for air vehicles. As a result of this development, a linear model of the plant at an arbitrary trim condition is easily obtained for design purposes.
- LQR design techniques utilizing a synthesis model and weighting "knobs" provide a straight forward means of obtaining satisfactory controller gains for MIMO systems while meeting design requirements.
- $\mathcal{D}$-Implementation of the linear trajectory tracking controller allows the controller to operate effectively on the nonlinear plant. In no-wind flight conditions, trajectories defined by an arbitrary $[v_0, \omega_0]$ are tracked perfectly in steady state. For flight conditions with wind, rejection of a constant wind disturbance is accomplished along the family of trajectories defined by an arbitrary $[v_0, \omega_0 = 0]$. However, for turning flight, a constant wind disturbance in $\{I\}$ is seen as a sinusoidal disturbance in $\{B\}$ and a sinusoidal tracking error results. For moderate bank angles and turn rates, the errors are usually small.

- Preprocessing of the trajectory commands by an adaptive filter allows for steady state control of the air vehicle's velocity with respect to the air mass in the forward path, thus not affecting stability. With sufficient margins for transient deviations in indicated airspeed, this would allieviate the major concern of stalling the air vehicle when tracking an inertial trajectory in wind.
- When analyzing a nonlinear plant and controller, test simulations are vital and in some cases the only means of performance evaluation. The three dimensional plots and time history graphs are fine for simple trajectories, but are difficult to analyze for more complex cases. The capabilities of a virtual prototyping software package, like Designer's Workbench, are impressive. The enhanced situational awareness and visualization capabilities of watching the designed controller operate on a virtual prototype allow for a "pilot's perspective" feedback not otherwise attainable on the desk top.

B. RECOMMENDATIONS

Based on the conclusions presented above and the experience of developing the simulation package presented in this thesis, the following recommendations are made.

- While the rigid body equations of motion are nonlinear with respect to the kinematics involved, they are completely linear with respect to the stability and control derivatives. The constant coefficient stability and control derivatives could be replaced by functions when further flight data is available.

- A similar design process used for the trajectory controller could be done using $\mathcal{H}_\infty$ design methodology.
- The trajectory preprocessor could be used to convert an inertial fixed trajectory into an air mass fixed trajectory. This might have applications where the air vehicle's inertial position is of secondary importance compared to its performance with respect to the air mass.
- Running simulations real time in Desiner's Workbench rather than using batch post processed data would be the next logical step. Further work might lead to virtual prototype visualizations based on real-time simulations or downlinks from actual air vehicles.

APPENDIX A: MATLAB FILES

The SIMULINK models of Bluebird (*plant1.m & plant2.m*) use the following *MATLAB* .m files as user defined functions.

STATE_DERIV.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                              %
%        Function to calculate derivative of u,v,w,p,q,r       %
%        based on                                              %
%                 1: kinematics                                %
%                 2: gravity                                   %
%                 3: propulsion                                %
%                 4: aerodynamics                              %
%                                                              %
%        Variables brought from workspace:                     %
%                                                              %
%        x = [contrl inputs, state variables(1 - 9), wind vel] %
%          = (da,de,dr,dtrt,u,v,w,p q,r,phi,theta psi, wind xyz)%
%                                                              %
%        Variables called from function "blue_data"            %
%                                                              %
%               rho = air density                              %
%               b = wing span                                  %
%                 c = wing cord                                    %
%                 s = wing area                                    %
%                 Cfo = Steady state force term                    %
%                 Cfu = Stability derivative for control inputs    %
%                 m = airplane mass                                %
%                 Ib = inertia tensor matrix (body frame)          %
%                 To = Thrust scale term                           %
%                 Pe = Engine postion matrix                       %
%                                                                  %
%                                                                  %
%                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function accel = state_deriv(x)

%%%%%% Function call to get the aircraft data

[u0,w0,rho,Cfx,Cfo,Cfu,Cfxdot,s,b,c,m,Pe,To,Ib] = blue_data;

%%%%%% seperate the combined vector into seperate elements

u = [x(1); x(2); x(3)];
dtrt = x(4);
state = [x(5); x(6); x(7); x(8); x(9); x(10)];
lambda = [x(11); x(12); x(13)];
```

```
wind = [x(14); x(15); x(16)];


%%%%%%  calculate velocity wrt the airmass and form state vector
%%%%%%  that will be used to calculate the aerodynamic forces/moments

ias = u + wind
state1 = [ias(1)-u0; ias(2); ias(3)-w0; x(8); x(9); x(10)];

%%%%%%  calculate total velocity, vt

vt = (ias(1)^2 + ias(2)^2 + ias(3)^2)^.5;

%%%%%%   calculate qbar

qbar = .5*rho*(vt^2);

%%%%%%  calculate M1

M1 = diag([1/vt, 1/vt, 1/vt, (.5*b)/vt, (.5*c)/vt, (.5*b)/vt]);

%%%%%%  calculate M2

M2 = diag([0, 0, (.5*c)/(vt^2), 0, 0, 0]);

%%%%%%   calculate Sprime
```

```
Sprime = diag([-1, 1, -1, b, c, b]*s);

%%%%%%  calculate Mu

Mu = inv([eye(3)*m,zeros(3);zeros(3),Ib]);

%%%%%%   calculate Tw2b


alpha = ias(3)/vt;

beta = ias(2)/vt;

T1 = [cos(alpha), 0, -sin(alpha); 0,1,0; sin(alpha), 0, cos(alpha)];

T2 = [cos(beta), -sin(beta), 0; sin(beta), cos(beta), 0; 0,0,1];

Tw2b = [T1*T2, zeros(3); zeros(3), T1*T2];



%%%%%%   calculate Chi

Chi = eye(6) - Mu*Tw2b*qbar*Sprime*Cfxdot*M2;

%%%%%%   calculate Propulsion matrix

Prop = [ eye(3);

0,-Pe(3),Pe(2);

Pe(3),0,-Pe(1);

-Pe(2),Pe(1),0];
```

```
%%%%%%   calculate gravity vector and rotation matrix {I} to {B}

Rot = [1, 0, -sin(lambda(2));
0, cos(lambda(1)), cos(lambda(2))*sin(lambda(1));
0, -sin(lambda(1)), cos(lambda(2))*cos(lambda(1))];

Ru2b = [Rot;zeros(3)];

g = [0; 0; 32.174];
FgU = m*g;

%%%%%%   put the components due to gravity; thrust; and control surface
%%%%%%   deflections together for their contribution to x-dot

thrust = Prop*To*dtrt;
gravity = Ru2b*FgU;
ctrl = qbar*(Tw2b*(Sprime*(Cfo + (Cfu*u))));

xdotu = (Mu*(ctrl+thrust+gravity));

%%%%%%    calculate kinematic contribution

omegax = [0,-state(6),state(5);state(6),0,-state(4);-state(5),state(4),0];
wxIb = (-inv(Ib))*(omegax*Ib);
```

```
Rot = [-omegax, zeros(3); zeros(3), wxIb];

xdotrot = Rot*state;

%%%%%   state vector feedback component xdot

xdotcfx = qbar*(Mu*(Tw2b*(Sprime*(Cfx*(M1*state1)))));

%%%%%%  add three components of xdot together and premult by inv(Chi)

xdot= inv(Chi)*(xdotrot+xdotcfx+xdotu);


%%%%%%   return xdot


accel=xdot;
```

EULER_B2I.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                %
%       Transformation [p q r] to  lambda-dot                    %
%                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function ldot = eul_b2u(x)


%%%%%   seperate the composite vector 'x' into [p q r]
%%%%%   and [phi theta psi].


omega = [x(1); x(2); x(3)];
phi=x(4);
theta= x(5);
psi=x(6);


%%%%%   calculate the rotation matrix {I} to {B}
%%%%%   based on euler angles


Rb2u = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
0, cos(phi), -sin(phi);
0, sin(phi)*sec(theta), cos(phi)*sec(theta)];



%%%%%   calculate lamda-dot


ldot = Rb2u*omega;
```

<u>EULER_I2B.M</u>

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                                                                    %
%          Transformation lambda-dot  to  [p q r]                    %
%                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function omegadot = eul_i2b(x)

%%%%%   seperate the composite vector 'x' into lambda-dot
%%%%%   and [phi theta psi].

ldot = [x(1); x(2); x(3)];
phi=x(4);
theta= x(5);
psi=x(6);

%%%%%    calculate the rotation matrix {B} to {I}
%%%%     based on euler angles

Rb2i = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
0, cos(phi), -sin(phi);
0, sin(phi)*sec(theta), cos(phi)*sec(theta)];

%%%%    calculate lamda-dot

omegadot = inv(Rb2i)*ldot;
```

POSITION_B2I.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%       From the workspace:                                     %
%                                                               %
%               1: free vector 'u' resolved in {B} e(1:3)       %
%               2: euler angle vector {phi,theta,psi} e(4:6)    %
%                                                               %
%       Returns:                                                %
%                                                               %
%               1: free vector 'u resolved in {I}               %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ans = pos_b2i(e)


%%%%%   this will rotate the trajectory error through phi, theta, psi
%%%%%   from {b} to {i}. (3-2-1 transformation


phi=e(4);
theta=e(5);
psi=e(6);
```

```
m_psi = [cos(psi),sin(psi),0
-sin(psi),cos(psi),0
0,0,1];

m_theta = [cos(theta),0,-sin(theta)
0,1,0
sin(theta),0,cos(theta)];

m_phi = [1,0,0
0,cos(phi),sin(phi)
0,-sin(phi),cos(phi)];


rotb2i = inv(m_phi*m_theta*m_psi);

u = [e(1); e(2); e(3)];

ans = rotb2i*u;
```

<u>POSITION_I2B.M</u>

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%       From the workspace:                                      %
%                                                                %
%               1: free vector 'u' resolved in {I} e(1:3)        %
%               2: euler angle vector {phi,theta,psi} e(4:6)     %
%                                                                %
%       Returns:                                                 %
%                                                                %
%               1: free vector 'u resolved in {B}                %
%                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ans = pos_i2b(e)


%%%%%   this will rotate through phi, theta, psi
%%%%%   from {i} to {b}. (3-2-1 transformation)


phi=e(4);
theta=e(5);
psi=e(6);


m_psi = [cos(psi),sin(psi),0
-sin(psi),cos(psi),0
0,0,1];


m_theta = [cos(theta),0,-sin(theta)
0,1,0
sin(theta),0,cos(theta)];


m_phi = [1,0,0
0,cos(phi),sin(phi)
0,-sin(phi),cos(phi)];


roti2b = (m_phi*m_theta*m_psi);


u = [e(1); e(2); e(3)];


ans = roti2b*u;
```

LIMIT_LOGIC.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%       funtion to limit trajectory commands, if required       %
%                                                               %
%               from workspace:                                 %
%                               1: commanded inertial velocity  %
%                               2: inertial wind                %
%                               3: lower IAS limit               %
%                               4: upper IAS limit               %
%                                                                %
%               returns:  revised commanded velocity             %
%                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function vcom=limit(u)


%%%%%  seperate u


vel_i = [u(1);u(2);u(3)];


wind_i = [u(4),u(5),u(6)];


ll=u(7);


ul=u(8);


%%%%%   calculate magnitud and direction of commanded velocity


gs=sqrt(vel_i(1)^2 + vel_i(2)^2);


ang=atan2(vel_i(2),vel_i(1));


%%%%%%   calculate commanded IAS (steady state)
```

```
vt= sqrt((vel_i(1)+wind_i(1))^2 +(vel_i(2)+wind_i(2))^2 +(vel_i(3)+wind_

%%%%%%    Prepare return variable (may not be limited)

vcom = vel_i;

%%%%%%    Check limits and revise if outside


if vt > ul;

over = vt - ul;

vcom(1) = (gs - over)*cos(ang);

vcom(2) = (gs - over)*sin(ang);
end;



if vt < ll;

under = ll - vt;

vcom(1) = (gs + under)*cos(ang);
vcom(2) = (gs + under)*sin(ang);

end;
```

BLUEBIRD_DATA.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%       Aircraft data for Bluebird                              %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [u0,w0,rho,Cfx,Cfo,Cfu,Cfxdot,s,b,c,m,Pe,To,Ib] = blue_data

%%%%%   trimmed flight speed and angle of attack

u0 = 73.3;

w0 = 0;

%%%%%%   Density: Sea level- std day

rho = .0023769;

%%%%%    derivative matrix due to state variables
```

```
%%%%%    rows: [CD CY CL Cl Cm Cn]
%%%%%    col:  [u v/U w/U p q r]


Cfx = [0 0 .188 0 0 0;
0 -.31 0 0 0 .0973;
0 0 4.22 0 3.94 0;
0 -.0597 0 -.363 0 .1;
0 0 -1.163 0 -11.77 0;
0 .0487 0 -.0481 0 -.0452];

%%%%%    derivative matrix due to control inputs

%%%%%    rows:  [CD CY CL Cl Cm Cn]
%%%%%    col:   [elev rud ail]

Cfu = [.065 0 0;
0 .0697 0;
.472 .0147 0;
0 .0028 .265;
-1.41 0 0;
0 -.0329 -.0347];

%%%%%     derivative matrix due to x-dot (alpha_dot & Beta_dot)
```

```
Cfxdot = [0 0 0 0 0 0;
0 0 0 0 0 0;
0 0 1.32 0 0 0;
0 0 0 0 0 0;
0 0 0 0 0 0];

%%%%%    steady state force vector

Cfo = [.03; 0; .385; 0; 0; 0];

%%%%%    physical dim.
%%%%%    WT =55 LBS.

m = 1.7095;
s = 22.38;
b = 12.42;
c = 1.802;

%%%%%    engine data (4 HP motor)

Pe = [0; 0; 0];
To = [15 ;0;0];

%%%%%    inertia tensor matrix

Ib = [ 10 0 0; 0 16.12  0; 0 0 7.97];
```

APPENDIX B: SIMULINK FILES

The nine state nonlinear model of bluebird is contained in the SIMULINK file *EOM_9.m* and was trimmed at a flight condition of

- flight speed equal to 73 fps
- flight path angle equal to zero

using the TRIM command. The resulting trim values for the state vector and input vector are:

$$x = \begin{bmatrix} 73.3 \\ 0 \\ -0.0023 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.2858 \end{bmatrix}$$

The LINMOD command was used to linearize the sixteen state nonlinear model of Bluebird (contained in the SIMULINK file *EOM_16.m* and described in Chapter III) about this trim point. The resulting linear system is contained in the MATLAB file *Linear16.mat.*

The rudder was removed as a control input (remove the second column of the B matrix) and the resulting linear model was used as a basis for the synthesis model contained in the SIMULINK file *synthesis.m.* This synthesis model was used to determine the LQR gain. The synthesis model, Q and R weighting matrices, and resulting LQR gain is contained in the MATLAB file *LQR_dat.mat.*

The full nonlinear simulation is contained in the SIMULINK file, *plant1.m.* The MATLAB .m file *simdata* loads the workspace with the appropriate variables. The file *simdata* calls the .m file *trajectory.m* in order to generate the trajectory schedule. Any changes to the commanded trajectory or wind disturbance schedule can be made in *trajectory.m.*

A version of the nonlinear simulation that does not use the filter and sensor blocks is contained in the SIMULINK file *plant2.m.* It runs considerably quicker than *plant1.m.*

APPENDIX C: $\mathcal{D}$-IMPLEMENTATION PROOFS REFERENCED

Identity. *Let $x \in R^3 = const$ be given. Then*

$$\frac{d}{d\Lambda}({}^{I}_{B}R(\Lambda)x) = -{}^{I}_{B}R(\Lambda)x \times S^{-1}(\Lambda). \tag{C.1}$$

and

$$\frac{d}{d\Lambda}({}^{B}_{I}R(\Lambda)x) = x \times {}^{B}_{I}R(\Lambda)S^{-1}(\Lambda). \tag{C.2}$$

Proof: To derive both equations we will need Poisson's Law:

$$\frac{d}{dt}({}^{I}_{B}R(\Lambda)) = {}^{B}\omega_B \times {}^{I}_{B}R(\Lambda), \tag{C.3}$$

and the following identity:

$$a \times b = -b \times a \tag{C.4}$$

for any vectors a and b of compatible dimensions. Now, consider

$$\begin{aligned} \frac{d}{dt}({}^{I}_{B}R(\Lambda)x) &= (\frac{d}{dt}({}^{I}_{B}R(\Lambda))x + {}^{I}_{B}R(\Lambda)\frac{d}{dt}x \\ &= {}^{B}\omega_B \times {}^{I}_{B}R(\Lambda)x = -{}^{I}_{B}R(\Lambda)x \times {}^{B}\omega_B, \end{aligned} \tag{C.5}$$

using equation (C.3), (C.4) and $x = const$.

Next, by the chain rule we get

$$\begin{aligned} \frac{d}{dt}({}^{I}_{B}R(\Lambda)x) &= \frac{d}{d\Lambda}({}^{I}_{B}R(\Lambda)x)\frac{d}{dt}\Lambda \\ &= \frac{d}{d\Lambda}({}^{I}_{B}R(\Lambda)x)S(\Lambda)\,{}^{B}\omega_B. \end{aligned} \tag{C.6}$$

Equation (C.1) now follows by comparing equations (C.5) and (C.6).

To obtain equation (C.2), consider

$$\frac{d}{dt}({}^{I}_{B}R(\Lambda){}^{B}_{I}R(\Lambda)) = \frac{d}{dt}({}^{I}_{B}R(\Lambda)){}^{B}_{I}R(\Lambda) +{}^{I}_{B} R(\Lambda)\frac{d}{dt}({}^{B}_{I}R(\Lambda)) = 0, \qquad \text{(C.7)}$$

since ${}^{I}_{B}R(\Lambda){}^{B}_{I}R(\Lambda) = I \ \ \forall\Lambda$. Now, using equations (C.3) and (C.7) we get:

$$\frac{d}{dt}({}^{B}_{I}R(\Lambda)) = -{}^{B}_{I}R(\Lambda)^{B}\omega_{B} \times . \qquad \text{(C.8)}$$

Finally, following the steps in the derivation of equation (C.1) we obtain:

$$\frac{d}{d\Lambda}({}^{B}_{I}R(\Lambda)x) = x \times{}^{B}_{I} R(\Lambda)S^{-1}(\Lambda).$$

■

Theorem ..1 *Suppose that assumptions A1 through A3 hold.*

A1. $Dim(x_{c2}) = dim(u) = dim(y_2)$.

A2. The matrix

$$\begin{bmatrix} sI - A_{c1} & B_{c3} \\ -C_{c1} & C_{c2} \end{bmatrix}$$

has full rank at $s = 0$.

A3. The matrix pair (A_{c1}, C_{c1}) *is observable.*

Then for each equilibrium point of $\mathcal{G}$ *in* $\mathcal{E}$ *the following properties are observed:*

- *the feedback systems* $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{K}\ )(r_0)$ *and* $\mathcal{T}(\mathcal{G}_{l_0}\ ,\ \mathcal{K}_l\ )$ *have the same closed loop eigenvalues;*

-

$$\begin{aligned} \mathcal{T}_l(\mathcal{G}\ ,\mathcal{K}\ )(r_0)(s) = \quad & L(\Lambda_0)\mathcal{T}(\mathcal{G}_{l_0}\ ,\ \mathcal{K}_l\ )(s)L^{-1}(\Lambda_0), \\ & \Lambda_0 = [0\ I]x_{p_0} \end{aligned}$$

Proof: In the proof we set the controller matrices D_{c1}, D_{c2} D_{c3} to zero. This does not change the results but considerably simplifies the algebra. Furthermore, we will drop explicit dependence of the controller parameters on α. Let $(x_{v_0}, x_{p_0} = \begin{bmatrix} P_0 \\ \Lambda_0 \end{bmatrix}, u_0, r_0 \in \mathcal{E})$ be given. Consider the feedback interconnection of the linear plant $\mathcal{G}_{l_0}\,(\Lambda_0)$ and linear controller $\mathcal{K}_l$. The state matrix F of this feedback system has the following form:

$$F := \begin{bmatrix} A_1 & \bar{A}_2 & B_2C_{c1} & B_2C_{c2} \\ I & \bar{A}_4 & 0 & 0 \\ B_{c1}C_1 & 0 & A_{c1} & B_{c3} \\ C_2 & 0 & 0 & 0 \end{bmatrix}. \tag{C.9}$$

Next we linearize the feedback interconnection of the plant $\mathcal{G}$ and the controller $\mathcal{K}$. However, in order to that, first we must determine the values of the controller states x_{c1} and x_{c2} along the trajectory $r_0 \in \mathcal{E}$. From equation (4.9) we obtain:

$$\begin{aligned} &\frac{d}{dt}x_{c1_0} = A_{c1}x_{c1_0} + B_{c1}\frac{d}{dt}y_{1_0} + B_{c3}L^{-1}(\Lambda_0)e_0 \\ &\frac{d}{dt}x_{c2_0} = C_{c1}x_{c1_0} + C_{c2}L^{-1}(\Lambda_0)e_0 \\ &u_0 = x_{c2_0}. \end{aligned}$$

Notice, since along r_0:

$$e_0 = 0, \quad y_{1_0} = const, \quad x_{c2_0} = u_0 = const$$

we get

$$\begin{aligned} &\frac{d}{dt}x_{c1_0} = A_{c1}x_{c1_0} \\ &0 = C_{c1}x_{c1_0}. \end{aligned}$$

Now, using Assumption $A3$ we conclude that $x_{c1_0} = 0$.

In order to compute the linearization of the feedback interconnection of $\mathcal{G}$ and $\mathcal{K}$, we must first obtain the linearizations of equations (4.3) and (4.9) about the operating points $(x_{v_0}, x_{p_0}, u_0) \in \mathcal{E}$ and $(x_{c1_0} = 0, x_{c2_0} = u_0, \dot{y}_{1_0} = 0, e_0 = y_{2_0} - r_0 = 0)$ respectively, determined by $r_0 \in \mathcal{E}$. The linearization of the plant $\mathcal{G}$ is given by (4.4). The linearization of the controller $\mathcal{K}$ has the following form:

$$\begin{aligned} \dot{\xi}_{c1} &= A_{c1}\xi_{c1} + B_{c1}\dot{\theta}_1 + B_{c3}L^{-1}(\theta_2 - \rho) \\ \dot{\xi}_{c2} &= C_{c1}\xi_{c1} + C_{c2}L^{-1}(\theta_2 - \rho) \\ \eta &= \xi_{c2}. \end{aligned} \tag{C.10}$$

It is easy to verify that the state matrix M of $\mathcal{T}_l(\mathcal{G}, \mathcal{C})(r_0)$ is

$$M := \begin{bmatrix} A_1 & A_2 & 0 & B \\ L & A_4 & 0 & 0 \\ B_{c1}C_1A_1 & B_{c1}C_1A_2 + B_{c3}L^{-1} & A_{c1} & B_{c1}C_1B \\ 0 & C_{c2}L^{-1} & C_{c1} & 0 \end{bmatrix}. \tag{C.11}$$

Let

$$P := \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & L & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}.$$

Obviously,

$$P^{-1} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & L^{-1} & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}.$$

Now using simple algebra it is easy to show that

$$\begin{aligned} \bar{M} &= P^{-1}MP \\ &= \begin{bmatrix} A_1 & \bar{A}_2 & 0 & B \\ I & \bar{A}_4 & 0 & 0 \\ B_{c1}C_1A_1 & B_{c1}C_1\bar{A}_2 + B_{c3} & A_{c1} & B_{c1}C_1B \\ 0 & C_{c2} & C_{c1} & 0 \end{bmatrix} \end{aligned}$$

$$
= \left[\begin{array}{c|c|c}
\begin{matrix} A_1 & \bar{A}_2 \\ I & \bar{A}_4 \end{matrix} & \begin{matrix} 0 \\ 0 \end{matrix} & \begin{matrix} B \\ 0 \end{matrix} \\
\hline
B_{c1} \begin{bmatrix} C_1 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A_1 & \bar{A}_2 \\ I & \bar{A}_4 \end{bmatrix} + B_{c3}[0 \;\; I] & A_{c1} & B_{c1}C_1B \\
\hline
0 \quad C_{c2} & C_{c1} & 0
\end{array}\right]
$$

The proof of the first part of the theorem now follows from Assumption $A2$ and an observation that the matrices F and $\bar{M}$ are in the form of the matrices F and M. The proof of the second part of the theorem consists of the following steps:

1. compute the transfer matrix of the linearization of the controller $\mathcal{K}\,(\Lambda)$ using equation (C.10) from controller inputs θ_1, θ_2, ρ to controller output η;
2. apply a similarity transformation
$$P_1 = \begin{bmatrix} I & 0 \\ 0 & L^{-1} \end{bmatrix}.$$
to the linearization of the plant $\mathcal{G}\ (= \mathcal{G}_l\,(r_0))$ given by equation (4.4) and derive the transfer matrix from the control inputs of the linear plant η to the outputs θ_1 and θ_2 using this new state-space realization;
3 compute the feedback interconnection of the transfer matrices obtained in steps 1 and 2 to get the final result.

A simple computation shows that the transfer matrix from the controller inputs θ_1, θ_2, ρ to the controller output η is given by:

$$
\begin{aligned}
\hat{\eta}(s) &= C_{c1}(sI - A_{c1})^{-1}(B_{c3}L^{-1}\frac{I}{s}(\hat{\theta}_2(s) - \hat{\rho}(s)) \\
&+ B_{c1}\hat{\theta}_1(s) + C_{c2}L^{-1}\frac{I}{s}(\hat{\theta}_2(s) - \hat{\rho}(s)) \\
&= K_l(s) \begin{bmatrix} P_1 \begin{bmatrix} \hat{\theta}_1(s) \\ \hat{\theta}_2(s) \end{bmatrix} \\ L^{-1}\hat{\rho}(s) \end{bmatrix},
\end{aligned}
\tag{C.12}
$$

where $K_l(s)$ is the transfer matrix for the controller $\mathcal{K}_l$.

Applying similarity transformation P_1 to equation (4.4) and computing the transfer matrix from η to θ_1 and θ_2 results in:

$$\begin{bmatrix} \hat{\theta}_1(s) \\ \hat{\theta}_2(s) \end{bmatrix} = \left[\begin{array}{cc|c} A_1 & \bar{A}_2 & B \\ I & \bar{A}_4 & 0 \\ \hline C_1 & 0 & D_1 \\ 0 & L & 0 \end{array}\right] \hat{\eta}(s) \tag{C.13}$$

where the transfer matrix is given in packed matrix form. A simple observation shows that

$$\begin{bmatrix} \hat{\theta}_1(s) \\ \hat{\theta}_2(s) \end{bmatrix} = P_1^{-1} G_{rb_l}(s)\hat{\eta}(s),$$

where $G_{rb_l}(s)$ is the transfer matrix for the plant $\mathcal{G}_{l_0}$.

Now routine algebra shows that the transfer matrix from ρ to θ_2 of the feedback interconnection of the transfer matrices in equations (C.12) and (C.13) is given by:

$$T_l(\mathcal{G}\ ,\mathcal{K}\ )(r_0)(s) = L(\Lambda_0)T(\mathcal{G}_{l_0}\ ,\ \mathcal{K}_l\ )(s)L^{-1}(\Lambda_0).$$

■

REFERENCES

1. Kuechenmeister, David R., "A Non-Linear Simulation For An Autonomous Unmanned Aerial Vehicle," Master's Thesis, Department Of Aeronautics, Naval Postgraduate School, Monterey, CA, September 1993.
2. Marquis, Carl, " Integration of Differential GPS and Inertial Navigation using Complementary Kalman Filter, "Master's Thesis, Department Of Aeronautics, Naval Postgraduate School, Monterey, CA, September 1993.
3. Kaminer, I.I., Khargonekar, P.P., and Robel, G., "Design of Localizer Capture and Track Modes For a Lateral Autopilot Using H_∞ Synthesis," IEEE Control Systems Magazine, vol 10, pp. 13-21, 1990.
4. Roskam, Jan, and Lan, Edward, *Airplane Aerodynamics And Performance,* Roskam Aviation And Engineering Corp., 1980.
5. Craig, J.J., *Introduction to Robotics Mechanics and Control,* Addison-Wesley, New York, 1986.
6. Schmidt, L.V., *Class Notes for AE3340,* U.S. Naval Postgraduate School, Monterey, CA. 1992.
7. Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls,* Roskam Aviation and Engineering corp, Ottawa, KS, 1979.
8. Watson Industries, *Techical Specifications for IMU–600D,* Watson Industries, Eau–Claire, WI, 1993.
9. Kaminer, I.I., *Class Notes For AE4276,* U.S. Naval Postgraduate School, Monterey, CA. 1993.
10. Kaminer, I.I., Pascoal, A.M., and Khargonekar, P.P., "A Velocity Algorithm For Implementation Of Non-Linear Gain-Scheduled Controllers, " submitted for publication in Automatica.
11. Nelson, R,C, *Flight Stability and Automatic Control,* McGraw-Hill, Inc., 1989.
12. Kwakernaak, H., and Sivan, R., *Linear Optimal Control Systems,* John Wiley and Sons, 1972.
13. Lagier, Mark, "An Application of Virtual Prototyping to the Flight Test and Evaluation of an Unmanned Air Vehicle, " Master's Thesis, Department Of Aeronautics, Naval Postgraduate School, Monterey, CA, March 1994.
14. Lagier, Mark *Virtual Prototype Demonstration Tape,* U.S. Naval Postgraduate School, Monterey, CA. March, 1994.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Isaac I. Kaminer Department of Aeronautics and Astronautics, Code AA/KA Naval Postgraduate School Monterey, CA 93943-5000	5
4.	Dr. Richard M. Howard Department of Aeronautics and Astronautics, Code AA/Ho Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Chairman Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, CA 93943-5000	2
6.	Eric L. Hallberg 3328 Susquehanna Road Dresher, PA 19025	1
7.	Richard Kershaw 246 Glenbrook Lane Greenville, SC 29615	1

	No. of Copies
8. Eric N. Hallberg 372 F Bergin Drive Monterey, CA 93940	2

DUDLEY KN
NAVAL POSTG OOL
MONTEREY CA 9394

GAYLORD S

3 2768 00307131 7